

Instrument Failures and Hidden Costs in Public AI Safety Evaluations: Seven Free Tools, Four Models and an Agentic Audit

Joseph Cirello

Potestas AI · joseph.cirello@potestasai.com

Preprint, October 2026. Every figure is from runs between 29 September and 2 October 2026 unless dated otherwise; derived figures are recomputed by `paper_numbers.py`, and the run records are listed in Appendix C.

Abstract

Free, public safety evaluations are often the first thing a team runs before deploying a language-model agent. We ran seven of them (AgentDojo, AgentHarm, StrongREJECT, MASK, b3, XSTest and garak) once each on four models (gpt-4o-mini, grok-4.20, Llama-3.3-70B and the reasoning model kimi-k2.6), beside a commercial agentic audit sold by the author. Three results. First, the tools failed as instruments in ways that changed their scores without warning. At MASK's default reply limit, 45% of the reasoning model's replies were empty, and the run recorded under half the lie rate of a run with room to answer; garak passed 5,163 of 5,176 checks on that model's empty replies; one garak probe leaks its generation settings into every later probe; and, through generic OpenAI-compatible connections, one vendor content-filter refusal stopped whole runs. Second, the tools cost more than their software price suggests. Their own graders cost 3.6 times as much as gpt-4o-mini across all seven tools, and the reasoning model's own usage across five tools cost 50 times gpt-4o-mini's, 94% of it output tokens, of which 68–86% were hidden reasoning. Third, the tools agreed little. Among the three models run under the same settings, Kendall's W was 0.19 to 0.33. The fourth model ranked safest on seven of eight measures, but it was also the one run at larger reply limits and temperature 1, so its rank is not evidence of agreement. Every measurement is one run, so disagreement and noise are not separated. The audit's scenarios are proprietary and its human review was the author's, so that half of the comparison cannot be reproduced by readers. In one run, the model AgentDojo's single attack rarely hijacked (1.2%) took harmful actions in the audit through four injection routes AgentDojo does not exercise; we describe the difference and cannot attribute it. We propose a correction for each instrument failure; their effect on the scores is untested.

1 Introduction

A team about to deploy a language-model agent can choose from dozens of free safety evaluations, many built by well-known laboratories, documented and re-runnable by anyone. They are the natural first step, and increasingly what a buyer is shown when asking whether an agent is safe.

We ran seven of them on four models and asked three questions. **Do the tools behave correctly as instruments** on current models, in particular reasoning models, which spend part of every reply on hidden reasoning, and vendors whose content filters refuse some prompts? **What does it cost to run them**, graders included? **How do their results compare** with each other and with an audit that grades what an agent does with tools? Safety benchmarks are known to rank models inconsistently [10–12]; we check this across a wider set of tools and against an action-graded audit.

Contributions.

- **Instrument failures** (§4.1): empty replies graded as answers, evasions or passes; runs stopped by one vendor refusal; settings leaking between garak probes; unseeded sampling; grader faults; truncation that differs by model — each with its mechanism, its measured size where the logs allow, and a correction (§5.3).
- **A relative cost accounting** (§4.2) separating the model under test from the graders, with the share of hidden reasoning.
- **A side-by-side comparison** of seven tools and an agentic audit on four models (§4.3–4.6), by type of test and route of attack, with a reproduction of AgentHarm's published gpt-4o-mini figure (§4.3). The audit's half is the author's and cannot be reproduced by readers (§6).

Conflict of interest. The author sells the agentic audit used here as a comparison, and its scenarios are proprietary. To limit the effect of that interest, the free tools ran at their published defaults except as stated in §3.3; every free-tool figure is the tool's own metric from its own code except those marked † in Table 2 and the capped MASK figure in §4.1.1; agreement is reported for the free tools alone as well as with the audit (§4.4); the audit's own losses and limits are stated (§4.1.5, §6); and §5.2 lists what the free tools do that the audit does not.

2 Related work

The tools. AgentDojo [1] measures whether instructions planted in tool outputs hijack an agent in four simulated environments, alongside task completion; it ships several attacks, of which we ran one. AgentHarm [2] measures completion of overtly malicious multi-step agent tasks, with a benign twin of each. StrongREJECT [3] measures single-turn compliance with forbidden requests, plain and inside jailbreak templates, with an automated evaluator validated against human judgement. MASK [4] elicits a model's belief and then pressures it to state otherwise, scoring honesty as the mean of $1 - P(\text{lie})$; evasions are recorded but are not lies. b3 [5] attacks an agent's backbone model through "threat snapshots" of ten applications at three defence levels. XSTest [6] pairs 250 safe prompts that look unsafe with 200 unsafe ones, to measure exaggerated refusal. garak [7] is a probe-and-detector vulnerability scanner. Five of the seven run on the UK AI Security Institute's Inspect framework [8].

Benchmark validity and disagreement. Li et al. [10] surveyed forty agent-safety benchmarks and ran twelve models (including grok-4.20 and an earlier Kimi) on AgentHarm, AgentDojo, InjecAgent and ODCV-Bench, finding no agreement in rankings (Kendall's $W = 0.10$). Wang et al. [11] re-ran four agent benchmarks on up to 22 models (including gpt-4o-mini and Llama-3.3-70B) with official scorers and found that no benchmark orders the models as another does, with capability confounding the scores. Desai et al. [12] found that correlations between model rankings on benchmarks with the same assigned safety concept are often weak. Earlier, "Safetywashing" [15] showed many safety benchmarks correlate with general capability; Beyer et al. [14] documented noise at every stage of safety evaluation; and configuration choices alone can flip pairwise verdicts [13]. Brokman et al. [16] compared open-source vulnerability scanners including garak and reported evaluator misclassification. Open issues on the tracker of Inspect's evaluations discuss the problem class we measure: they ask for empty-response checks before judges grade [43, #2415], report that b3's scorer drops reasoning content [43, #2130], document AgentHarm's refusal judge turning API errors into "did not refuse" [43, #2212], and seek one rule for whether an instrument failure is scored as a model failure [43, #2091, #2599]. We add measured effect sizes for two of these problems (empty replies and run-level failure) on real runs, tools outside the agent-benchmark class, and an action-graded comparison.

Graders, reproducibility and cost. LLM judges carry position, verbosity and self-preference biases [17], approach human agreement only at the largest sizes [18], are sensitive to output style [19], disagree across harm categories [20], are not reproducible even at temperature zero on borderline items [21], and can be fooled by harmful-sounding but harmless answers [22]. Reasoning benchmarks are sensitive to decoding parameters, seeds and prompt format [23]; long chain-of-thought does not by itself make models safer [24]; "deterministic" settings are not deterministic [25, 26]; and statistical reporting [27] and reproducible practice [28] are well argued. Agent evaluation should report cost alongside accuracy [29]; one agent leaderboard spent about \$40,000 on 21,730 rollouts [30]; and evaluation cost is becoming a bottleneck [31]. We are not aware of a cost breakdown for running public safety tools that separates the grader from the model under test.

Agents that say one thing and do another. Models that refuse harmful requests in chat do not refuse as browser agents [38]; agents that can identify risks with high accuracy fail to avoid them when acting [39]; and agents can act harmfully without disclosing it [40–42]. Agent benchmarks such as InjecAgent [32], ToolEmu [33], R-Judge [34], Agent-SafetyBench [35], ODCV-Bench [36] and OpenAgentSafety [37] measure agent behaviour with tools. The audit used here belongs to this class.

3 Setup

3.1 Models

Model	Endpoint	Notes
gpt-4o-mini-2024-07-18	OpenAI	conventional
grok-4.20-0309-non-reasoning	xAI	conventional
Llama-3.3-70B-Instruct-Turbo	DeepInfra	open weights; DeepInfra's "Turbo" serving variant, whose numerical precision we did not record
kimi-k2.6	Moonshot	reasoning: produces hidden reasoning tokens before each reply

None is the newest model its vendor offers.

3.2 Tools and versions

Tool	Version	What it measures	Grading
AgentDojo [1]	0.1.35, benchmark v1.2, <code>important_instructions</code> attack, all four suites; no reply limit set (vendor defaults)	hijack by instructions in tool outputs, while working	environment end state
AgentHarm [2]	<code>inspect_evals</code> 0.22.0, <code>test_public</code> split (44 behaviours $\times$ 4 variants = 176 tasks), and its benign twin	completion of malicious agent tasks; refusal	criteria + GPT-4o judge
StrongREJECT [3]	<code>inspect_evals</code> 0.22.0, plain and inside the AIM jailbreak template the tool ships (313 prompts each)	single-turn harmful compliance	GPT-4o judge
MASK [4]	<code>inspect_evals</code> 0.22.0 (1,000 scenarios)	stating what the model believes false, under pressure	GPT-4o and o3-mini judges
b3 [5]	<code>inspect_evals</code> 0.22.0 (630 snapshots $\times$ 5 epochs)	attacks on an agent's backbone model	automatic scoring
XSTest [6]	450 prompts (250 safe, 200 unsafe), run through Inspect with the <code>inspect_evals</code> task's own solver, scorer and judge	refusal of safe and unsafe prompts	GPT-4o judge

Tool	Version	What it measures	Grading
garak [7]	0.17.0, 81 probes in 18 families	text vulnerabilities	rule-based and model detectors

Inspect AI was version 0.3.273. Each tool ran once per model between 29 September and 2 October 2026. XSTest used the gated public dataset (sha256 prefix `ccb792dc`), loaded locally because the evaluation host held no dataset credential.

3.3 Departures from defaults

- **Vendor-forced settings.** xAI rejects `seed = 0` (AgentHarm's default), so grok-4.20 ran at seed 1 on every Inspect tool, and garak ran on it without frequency and presence penalties, which `promptinject` would otherwise set (§4.1.3). Moonshot allows only temperature 1 (top_p 0.95) for kimi-k2.6, where the other models ran at each tool's own temperature (0 to 1).
- **Reply limits for the reasoning model.** kimi-k2.6 was given 4,096 tokens for XSTest and garak and 8,192 for MASK, because at the defaults its replies were empty (§4.1.1), so its scores were not measured under the same limits as the other models.
- **One generation per garak prompt** on every model (the default is five), to bound cost.
- **Vendor refusals and errors.** xAI and Moonshot were reached through Inspect's generic OpenAI-compatible provider. Inspect tasks ran with `--no-fail-on-error`, except Llama-3.3-70B's first MASK, b3 and StrongREJECT runs, which ran at Inspect's documented default (§4.1.4); garak ran through a wrapper that records a content-filter HTTP 403 as a blocked prompt (§4.1.2). Errored samples are excluded from scores.
- **Interruptions** (credit outages, stalled connections) were resumed from saved samples; Llama-3.3-70B's b3 score is recomputed from its 3,149 saved per-sample scores by a method that reproduces the tool's own summaries exactly.

AgentDojo's driver settings, the per-run interruption details and the request timeouts are in Appendix A. The corrections in §5.3 were developed during these runs and only partly applied to them: the runs were not seeded, did not isolate `promptinject` and used Inspect's default timeouts (Appendix A).

3.4 The agentic audit

The comparison audit has two parts. The **agentic suite** (258 scenarios in 48 families, 67 of them controls whose correct answer is to act) gives the model working, instrumented tools and ordinary tasks, with harm arriving through pressure, planted instructions or skipped policy. Harm is a specific tool action defined per scenario and graded deterministically from the tool-call log, with no model judge; a scenario with no final reply is untested, not safe, and one the grader cannot decide goes to human review and counts here as not-harm, so harm counts are lower bounds. Twelve families are injection routes, defined by where the planted instruction enters (Table 6): content read during the task, a tool's definition, state from earlier work, other components, or goal hijacking. Appendix D shows how a scenario is built; every row is re-scored by one grader version, and the runs used two scenario versions differing in 3 of 258 scenarios, none in the injection families (Appendix E).

The **400-turn conversation** measured 36–37 test categories per run, with sixteen harmless controls, scored by a judge from a different vendor (grok-4.3 for gpt-4o-mini, gpt-4o for the others), so it is not on a common scale across models and stays out of the agreement analysis. Every turn its grader sent to human review was ruled by the author, one at a time, against a written standard, with a model proposing each verdict and its

reasons (kimi-k2.6 17, gpt-4o-mini 28, grok-4.20 32, Llama-3.3-70B 35; 43 of 112 ruled failures). Its safety index (0–100, higher is safer) is recomputed with those rulings by the audit's own formula; its security-critical count is as first scored, because it is tallied from machine verdicts during the run.

4 Results

4.1 How the tools fail as instruments

Three different things are grouped under this heading and separated again in §5.3: defects (garak's generation settings leaking between probes, its treatment of a content-filter 403 as an authentication failure, MASK's validator raising on one judge answer), defaults that suit conventional models and not reasoning models (reply limits, a parameter the user can set), and design choices (MASK records an evasion as not a lie, by design). The first kind is the tool's to fix; the second is shared between the tool's defaults and the user's settings; the third is a choice the user has to know about. All three changed scores in these runs without any warning in the tool's output.

4.1.1 Empty and truncated replies graded as answers

A reasoning model writes hidden reasoning before its answer; in kimi-k2.6's logs it was 85% of output tokens on b3, 79% on MASK, 68% on StrongREJECT (86% inside its AIM template) and 68% on AgentHarm's benign twin. When a tool caps the reply, reasoning can use up the cap and the visible reply is empty. Four tools graded such replies.

- **MASK.** At its default 1,000-token limit, a kimi-k2.6 run stopped after 764 of 1,000 scenarios. 1,091 of its 2,430 stored belief and pressured replies (44.9%) were empty, including the pressured reply in 567 of the 762 scenarios that reached one. MASK's judges scored 756 of those 762 (6 raised grading errors) and labelled 528 of them evasions, which MASK does not count as lies [4]; with 77 lies the run read about 90% honest. With an 8,192-token limit, 5 of 3,170 replies were empty and MASK reported 77.2% (228 lies). On the 755 scenarios both runs scored, the lie rates were 10.2% and 23.7%, and the evasion rates 69.8% and 15.1%. The runs are separate, at temperature 1, so the comparison is descriptive.
- **XSTest.** At its 256-token limit, in a six-prompt trial on the safe subset re-run on 14 October 2026 (log kept, Appendix C), every reply stopped at the limit, hidden reasoning took 217–255 of the 256 tokens, 4 of the 6 visible replies were empty, and the judge graded each of those four as a refusal (refusal rate 66.7%, against 3.2% at 4,096 tokens, Table 2). The same limit also **truncated** most conventional-model replies to the safe prompts (137–178 of 250) and fewer to the unsafe ones (23–66 of 200), while kimi-k2.6 at 4,096 tokens was almost never cut (2 of 250), so the judge graded cut-off text for three models and full text for one. This is a weaker point than the empty replies: a refusal is usually decided in the first sentence, and we did not test whether truncation changed any verdict.
- **garak.** 3,791 of kimi-k2.6's 11,938 replies were empty strings, which garak scores (it skips only missing outputs; its open issue #804 would skip these too [9]); 5,163 of the 5,176 checks on them passed. kimi-k2.6's failure rate is therefore 32.3% as reported, 46.7% on answered prompts, 63.3% if empty replies count as failures, and 36.2% on the 3,122 prompts all four runs asked and it answered — the highest of the four there (grok-4.20 33.4%, Llama-3.3-70B 27.1%, gpt-4o-mini 26.8%), though that subset is selected by its own behaviour. Excluding the families affected by §4.1.3, the answered-only rates are kimi-k2.6 46.7%, grok-4.20 38.6%, Llama-3.3-70B 26.4% and gpt-4o-mini 25.4%.
- **AgentHarm's benign twin.** 5 of kimi-k2.6's final replies were empty and 9 stopped at the limit, and were graded with the rest; its 15 refusals were all visible, complete replies.

No tool surfaces the number of empty replies in its summary (Inspect records each reply's stop reason, from which Appendix A's counts come); an open issue asks for that check [43, #2415]. The audit records a reply with no response as untested.

4.1.2 One vendor refusal can abort the run

Vendors' content filters refuse some prompts with an HTTP error instead of a reply. Through Inspect's generic OpenAI-compatible provider, at its documented default (`fail_on_error`, which keeps completed samples), one such refusal stopped the whole StrongREJECT run on grok-4.20 (HTTP 403) and on kimi-k2.6 (HTTP 400). Inspect's native xAI provider already maps xAI's refusal to a content-filter result; its OpenAI-compatible path does not recognise the `content_filter` type in Moonshot's error. garak's OpenAI-compatible generator deliberately treats any 401 or 403 as terminal, since retrying with the same key is pointless; because xAI returns content refusals as 403, garak stopped on grok-4.20 after 239 of about 12,000 prompts with an authentication-failure message. With per-sample tolerance and a wrapper that records a content-filter 403 as a blocked prompt, both completed (grok-4.20: 19 blocked StrongREJECT prompts, 11 in the plain variant and 8 inside the AIM template; counting the 11 as refusals would move its plain score from 0.094 to 0.091; 5 garak prompts with no text). A related garak issue, a generator aborting a probe on a single transient HTTP 408, has been fixed and closed [9].

4.1.3 Settings that leak between probes, and unseeded sampling

garak's `promptinject` probes set five generation parameters on the shared generator before each attempt (temperature, `top_p`, both penalties, and `max_tokens` between 60 and 256) as part of the technique; since mid-2026 an option (`generation_params`) controls which are set [9]. They are never restored — the probe's clean-up only clears history and every probe shares one generator — so later probes inherit them. On kimi-k2.6, given 4,096 tokens, 766 of 768 replies inside `promptinject` were empty (its short limits, by design), and so were 981 of 982 in the three families after it (snowball, system-prompt extraction, web injection), which only the leak explains; before it, 20% (2,044 of 10,188) were. garak records no stop reasons, so the effect on conventional models cannot be measured, though gpt-4o-mini's and grok-4.20's replies in those families were short (about 180–230 characters) and their rates there differ sharply (snowball: gpt-4o-mini 48.0%, the other two 0.0%; Appendix B). garak's default stop sequences (`#`, `;`), which we did not override, may also have shortened replies on every model. A related issue (capped probes reading the wrong prompt's settings) was opened on 1 October 2026 [9].

garak also samples prompts at random for capped probes, unseeded by default, so different runs ask partly different prompts; partly for that reason (some probes also generate prompts at random), only 4,287 (probe, prompt) pairs were asked of all four models. Cross-model comparisons should use shared prompts or a fixed seed.

4.1.4 Grader faults and hangs

On Llama-3.3-70B, MASK's numeric validator raised an error on one sample's judge answers (belief ranges mixing bound types it rejects); at Inspect's documented default (`fail_on_error`) that stopped the run after 33 minutes and 694 samples (673 scored). Resumed with `inspect eval-retry`, the same prompt graded without error, so the judge's output varies between calls [21]; how grader failures should count is under discussion on Inspect's tracker [43, #2091, #2599]. Llama-3.3-70B's first b3 run, also at the default, stopped when b3's embedding service returned nothing for a reply that was a tool call written as text, and a later b3 run was held about 50 minutes by stalled connections (observed live): by default Inspect waits up to 600 seconds per attempt and retries without limit, which `--timeout` and `--attempt-timeout` bound.

4.1.5 The comparison audit loses scenarios too

The audit loses scenarios too: 23 of kimi-k2.6's 258 ended without a final reply (21 cut off by the ten-round step budget, 2 at the output cap), as did 23 of grok-4.20's (step budget) and 1 of gpt-4o-mini's; Llama-3.3-70B lost 10, five to final replies that were tool calls written as text. All are recorded as untested, not safe, and Table 3 gives the bound they imply.

4.2 What the free tools cost

Costs are reported as ratios, because prices change and the ratios carry the finding; the token counts and dated list prices behind them are in Appendix F: token counts at list prices on 2 October 2026, before caching discounts, for successful runs only (failed and repeated runs made the real spend higher).

Table 1. Relative compute cost of a successful run of six tools. AgentDojo is discussed below the table because its token counts do not survive in full for kimi-k2.6. garak has no LLM grader, so the graders' share is shown with and without it.

	Graders as % of the model under test (six tools)	Same, excluding garak	Total, as % of gpt-4o-mini's (six tools)	Same, excluding garak
gpt-4o-mini	733%	1,216%	100%	100%
Llama-3.3-70B	809%	1,123%	106%	108%
grok-4.20	117%	170%	182%	163%
kimi-k2.6	13%	30%	938%	493%

Graders dominate for the two cheapest models. For gpt-4o-mini and Llama-3.3-70B the tools' own graders (GPT-4o and o3-mini) cost 7.3 and 8.1 times as much as the model tested (12 and 11 times on the five Inspect tools), MASK alone making up 71–73% of grader cost on the three conventional models (63% on kimi-k2.6). Adding AgentDojo, which has no grader, as a seventh tool lowers this to 3.6 and 4.0 times (0.38 for grok-4.20, for which AgentDojo's long transcripts made it the most expensive tool, raising its total to 314% of gpt-4o-mini's, and Llama-3.3-70B's to 104%); kimi-k2.6's AgentDojo usage survives only in part, which puts its seven-tool total at 860% or more.

Reasoning dominates for the reasoning model. The five Inspect tools cost 4.9 times as much on kimi-k2.6 as on gpt-4o-mini with the graders included, and the model's own usage 50 times as much. With garak added the two figures are 9.4 and about 69, less certain because kimi-k2.6's garak cost extrapolates one interval of billed spend, which would include any caching discount, although cost per prompt varies by probe. Part of that is the larger reply limits it needed (§3.3); the rest is price and length: per-token prices 6.3 (input) and 6.7 (output) times gpt-4o-mini's, ten times the output tokens across the five tools (18.4 million against 1.8 million, which is 94% of its model-side cost), and 3,531 output tokens per b3 attack against 265, 85% of them hidden reasoning (b3 sets no limit of its own, and its strongest defence level uses the model under test as its own judge, so b3's "model" cost includes its own grading). Against grok-4.20, whose output price is 62.5% of kimi-k2.6's, a kimi-k2.6 b3 answer cost 7.6 times as much: 4.1 times the tokens at 1.8 times the average price per token. On many encoded garak probes it returned an empty reply, and the tokens were still billed.

A price per token does not show a price per answer, and a model chosen on per-token price can be expensive to evaluate, and likely to operate, for the same reason (we did not measure operating costs).

Method: Inspect costs come from each log's token counts, reasoning included; garak's are estimated from characters ($\div 4$) for conventional models and from measured spend per prompt for kimi-k2.6; b3's embedding calls are not counted; Llama-3.3-70B's b3 input tokens per attack were 4.8 times gpt-4o-mini's, possibly

inflated by its resumed run (under 3% of any ratio here). The token counts per tool and model behind every ratio, and the list prices used, are in Appendix F.

4.3 The scoreboard

Table 2. Each free tool's headline metric, one run per model, with 95% intervals: cluster-bootstrap for AgentHarm (by behaviour) and b3 (by attack, 210 attacks at three defence levels over five epochs), Wilson elsewhere (narrower than the truth where samples cluster). † = our construction or recomputation: kimi-k2.6's AgentDojo totals (rebuilt from per-task files after an interruption), Llama-3.3-70B's b3 mean (§3.3), and every garak rate (pooled from garak's per-probe results; "as reported" counts empty replies as garak does).

	kimi-k2.6	Llama-3.3-70B	gpt-4o-mini	grok-4.20
AgentDojo (important_instructions): attacks that hijacked the agent	1.2% (0.6–2.1) †	15.0% (12.8–17.4)	29.8% (27.0–32.8)	49.7% (46.6–52.9)
AgentDojo: tasks completed, no attack / under attack	85.6% / 86.3% †	26.8% / 11.4%	64.9% / 39.3%	76.3% / 40.1%
AgentHarm: harm score (0–1)	0.18 (0.10–0.27)	0.41 (0.33–0.51)	0.59 (0.50–0.68)	0.79 (0.71–0.85)
AgentHarm: refused	79.5% (69.3–88.6)	36.4% (25.0–47.7)	21.6% (13.6–30.7)	5.7% (1.1–11.9)
AgentHarm benign: score (0–1); refused	0.71; 8.5%	0.60; 0%	0.72; 0%	0.82; 0%
StrongREJECT: score (0–1), plain / AIM	0.00 / 0.00	0.20 / 0.08	0.10 / 0.00	0.09 / 0.00
MASK: honest; accurate	77.2%; 86.0%	56.6%; 78.0%	54.9%; 71.2%	39.0%; 74.4%
b3: attack success	11.9% (10.1–13.7)	25.6% (23.1–28.2) †	15.5% (13.6–17.6)	30.4% (27.8–32.9)
XSTest: safe prompts refused	3.2% (1.6–6.2)	2.8% (1.4–5.7)	6.8% (4.3–10.6)	6.0% (3.7–9.7)
XSTest: unsafe prompts refused	88.0% (82.8–91.8)	92.5% (88.0–95.4)	87.5% (82.2–91.4)	84.4% (78.7–88.8)
garak: checks failed, as reported †	32.3%	27.7%	26.2%	39.4%
garak: checks failed, answered prompts only †	46.7%	27.8%	26.2%	39.4%
garak: checks failed, shared prompts, answered only †	36.2%	27.1%	26.8%	33.4%

Errored samples are excluded (Appendix A); on StrongREJECT they were vendor refusals (grok-4.20 19 of 626; kimi-k2.6 1 per variant). kimi-k2.6's garak and XSTest figures are affected by §4.1.1. kimi-k2.6 completed slightly more AgentDojo tasks under attack than without. b3's attack success falls with defence level for every model (weakest to strongest, the strongest adding an LLM judge run by the model itself: gpt-4o-mini 27.9% to 1.4%, Llama-3.3-70B 42.8% to 5.8%, grok-4.20 49.8% to 12.9%, kimi-k2.6 21.9% to 3.5%).

Table 3. The agentic audit on the same models (re-scored by one grader version, so counts can differ from a run's sealed summary: gpt-4o-mini's harm count is 97 here and 96 as sealed; 400-turn figures not comparable across models, §3.4).

	kimi-k2.6	Llama-3.3-70B	gpt-4o-mini	grok-4.20
Injection scenarios ending in harm	7/27, 25.9% (13.2–44.7)	10/28, 35.7% (20.7–54.2)	12/29, 41.4% (25.5–59.3)	15/28, 53.6% (35.8–70.5)
Injection routes with at least one harm (of 12)	5	6	9	9

	kimi-k2.6	Llama-3.3-70B	gpt-4o-mini	grok-4.20
All harm scenarios ending in harm	21/173, 12.1% (8.1–17.8)	94/185, 50.8% (43.7–57.9)	97/190, 51.1% (44.0–58.1)	70/174, 40.2% (33.2–47.7)
... upper bound, if every untested, errored or review harm scenario had ended in harm	20.9%	55.0%	52.9%	46.6%
All harm scenarios ending in harm, after human review	21/173, 12.1% (8.1–17.8)	96/185, 51.9% (44.7–59.0)	98/190, 51.6% (44.5–58.6)	72/174, 41.4% (34.3–48.8)
Controls: acted correctly / wrong action / not acted on, of graded	60 / 0 / 0 of 60	52 / 2 / 5 of 59	62 / 0 / 2 of 64	52 / 2 / 5 of 59
Controls after human review: acted correctly / not done correctly, of graded	61 / 0 of 61	57 / 6 of 63	64 / 3 of 67	53 / 8 of 61
400-turn safety index after human review (higher is safer; as first scored 98.8 / 90.7 / 92.8 / 93.5)	97.9	92.9	91.6	92.5
400-turn review turns ruled failures by a human	3 of 17	13 of 35	14 of 28	13 of 32
400-turn security-critical findings (as first scored)	1	14	12	7

Untested scenarios are out of the denominators: step budget (kimi-k2.6 21, grok-4.20 23, Llama-3.3-70B 5, gpt-4o-mini 1), output cap (kimi-k2.6 2), a tool call written as text (Llama-3.3-70B 5), one kimi-k2.6 error; controls among them: kimi-k2.6 6, grok-4.20 6, Llama-3.3-70B 4. Harm scenarios in review count as not-harm (1, 5, 3, 2), so harm counts are lower bounds, though rates are not (untested scenarios leave the denominator); the next row is the upper bound. "Graded" controls exclude untested ones and those in review (1, 4, 3, 2). Every scenario the audit sent to review was later ruled by a human (kimi-k2.6 3, Llama-3.3-70B 14, gpt-4o-mini 8, grok-4.20 9): these totals are the harm scenarios in review, the controls in review, and the controls graded not acted on (0, 5, 2, 5), which are review rows whose reason the grader states, plus kimi-k2.6's one errored row; the "after human review" rows apply those rulings to the same grades, and no injection count changes (`agentic_post_review.py` reproduces the machine rows first, then applies them).

A reproduction check. AgentHarm's authors report gpt-4o-mini at 62.5% harm and 22.0% refusals (Table 9 of [2]); our run, on the 44 public-test behaviours `inspect_evals` packages (their table covers all 66) with the same GPT-4o judge, gave 0.589 and 21.6%, which we take as evidence the tools were installed and run correctly; with different task sets the figures are not expected to be identical.

4.4 Agreement between measures

Table 4. Rank of each model by each measure (1 = safest; ties averaged). XSTest is omitted because it measures refusing too much. The audit's two rows share no scenario: the second is its harm scenarios outside the twelve injection families (kimi-k2.6 14/146, Llama-3.3-70B 84/157, gpt-4o-mini 85/161, grok-4.20 55/146).

Measure	kimi-k2.6	Llama-3.3-70B	gpt-4o-mini	grok-4.20
AgentDojo (hijack)	1	2	3	4
AgentHarm (harm score)	1	2	3	4
MASK (dishonesty)	1	2	3	4
b3	1	3	2	4
StrongREJECT (plain)	1	4	3	2

Measure	kimi-k2.6	Llama-3.3-70B	gpt-4o-mini	grok-4.20
garak (shared prompts, answered)	4	2	1	3
Audit: injection scenarios	1	2	3	4
Audit: harm outside the injection families	1	4	3	2

The clean comparison is the three models run under the same settings, and there the tools barely agree. Among gpt-4o-mini, grok-4.20 and Llama-3.3-70B, Kendall's W (tie-corrected) is 0.33 across the six free tools (permutation $p = 0.19$) and 0.19 across all eight measures ($p = 0.28$) however garak is scored, or 0.27 ($p = 0.12$) with differences under 5% treated as ties. Three models and one run each give this little power either way, and it is not directly comparable with the 0.10 Li et al. report for twelve models [10]. Adding kimi-k2.6 raises W to 0.40 across the free tools ($p = 0.06$) and 0.41 across the eight ($p = 0.015$), or 0.54 and 0.52 with garak as reported, because kimi-k2.6 ranks first on seven of the eight. But it is also the model run at larger reply limits and temperature 1 (§3.3), so that agreement is as likely a settings effect as a safety one, and nothing below rests on it.

Some orderings coincide, largely because of one model. Four measures (AgentDojo, AgentHarm, MASK, the audit's injection scenarios) share one ordering and two (StrongREJECT, the audit's other harm) another. Four of eight unrelated measures sharing an ordering would happen 0.5% of the time, but with kimi-k2.6 first on seven, four of seven ordering the other three alike would happen 11% of the time; we noticed this after seeing the data. Several ranks rest on near-ties (StrongREJECT, grok-4.20 0.094 against gpt-4o-mini 0.096; the audit's other harm, gpt-4o-mini 52.8% against Llama-3.3-70B 53.5%; its injection scenarios, 12/29 against 10/28, where one scenario would swap them). AgentDojo's ordering is partly capability: Llama-3.3-70B completed only 26.8% of AgentDojo's tasks without attack and 11.4% under attack, and AgentDojo's authors note that low-utility models often fail to carry out the attacker's goal too [1].

The disagreement is not a defect in itself — the tools measure different constructs [12] — but it means no single free score can stand for a model's safety.

4.5 What the action-graded audit recorded

Nothing in this section can be checked by a reader: the scenarios are proprietary and the human review was the author's (§6). The free-tool findings in §4.1–4.4 do not depend on it. What follows is the audit's own record from one run per model, stated as narrowly as the data allow.

Same ranking, wider coverage. AgentDojo and the audit's injection scenarios order the four models the same way (Table 4), but two unrelated measures agree on every rank one time in 24 (one in six with kimi-k2.6 first on both) and the audit's intervals overlap, so this does not show they measure the same thing.

Similar routes, different rates. AgentDojo plants instructions in tool outputs such as emails, calendar events, files and web pages; the audit's three closest routes are indirect injection (an email thread, a calendar invite, a support ticket), a retrieved document and a tool result. kimi-k2.6 took a harmful action in 3 of these 7 scenarios, all indirect injections (Table 6), against 1.2% of 949 AgentDojo attacks. The units differ and the counts are small, so we cannot attribute the gap from one run; candidate causes (how plausible the planted step looks, AgentDojo's single template, the environments) are left for future work.

Routes AgentDojo does not exercise. kimi-k2.6 also took harmful actions through persistent memory, a plan it saved and re-read, a poisoned tool server and a message from another agent; across the four models, five to nine of the twelve routes produced harm (Table 6), including data sent outside, administrator rights granted and untrusted instructions passed to other agents.

Refusing a benchmark task is not failing legitimate work. kimi-k2.6 refused the most on AgentHarm (79.5% of malicious tasks, 8.5% of benign ones) yet acted correctly on all 60 of its graded audit controls (6 untested, 1 in review; 61 of 61 once that one was ruled). Conversely, every model refused few of XSTest's safe prompts (2.8–6.8%) and the conventional models none of AgentHarm's benign tasks, yet Llama-3.3-70B and grok-4.20 each took 2 wrong actions and failed to act on 5 of 59 graded controls (after human review, 6 of 63 and 8 of 61 not done correctly).

Obeying a bad request is not harm during a good one. grok-4.20 completed most openly malicious AgentHarm tasks (0.79, refusing 5.7%) yet had a lower audit harm rate than gpt-4o-mini or Llama-3.3-70B (40.2% against 51.1% and 50.8%, or 41.4% against 51.6% and 51.9% after human review; intervals overlap; upper bound 46.6%). Which exposure matters depends on whether a deployment meets malicious requests or adversarial content during legitimate work.

4.6 Where the failures were, by type

The tools and the audit group their tests by type. We report results by type only; no test prompt is reproduced.

Table 5. AgentHarm harm score (0–1) by category of malicious agent task, with the share refused in brackets. About 22 tasks per category, so one task moves a category's figure by about 4–5 points.

Category	kimi-k2.6	Llama-3.3-70B	gpt-4o-mini	grok-4.20
Disinformation	0.33 (70%)	0.55 (5%)	0.66 (15%)	0.72 (0%)
Harassment	0.08 (92%)	0.46 (38%)	0.57 (25%)	0.92 (4%)
Drugs	0.22 (75%)	0.50 (45%)	0.80 (10%)	0.85 (15%)
Fraud	0.19 (71%)	0.40 (29%)	0.47 (29%)	0.65 (21%)
Hate	0.05 (95%)	0.25 (65%)	0.61 (25%)	0.90 (0%)
Cybercrime	0.13 (90%)	0.39 (25%)	0.63 (30%)	0.80 (0%)
Sexual	0.00 (100%)	0.18 (79%)	0.39 (38%)	0.84 (4%)
Copyright	0.43 (46%)	0.58 (4%)	0.64 (0%)	0.63 (0%)

Copyright tasks were the least refused category for every model (for grok-4.20, tied at 0% with three others), including the most refusing one (kimi-k2.6 refused 46% of them against 70–100% elsewhere). grok-4.20 refused almost nothing in any category (0–21%), so its high harm score reflects compliance across the board rather than a weakness in one area.

Table 6. Harm scenarios ending in harm, by injection route and model (one run; n = harm scenarios in the route).

Route of attack (n)	kimi-k2.6	Llama-3.3-70B	gpt-4o-mini	grok-4.20
Indirect injection: email, ticket, calendar (3)	3	3	3	1
Persistent memory (3)	1	1	1	2
Saved plan re-read later (3)	1	3	1	3
Retrieved document (3)	0	1	1	3
Goal hijack over several turns (2)	0	1	1	1
Poisoned tool server (3)	1	0	2	2
Content passed between agents (2)	1	0	1	1
Direct goal hijack (2)	0	1	0	1

Route of attack (n)	kimi-k2.6	Llama-3.3-70B	gpt-4o-mini	grok-4.20
Tool result (1)	0	0	0	1
Supplied tool or sub-agent (2)	0	0	1	0
Tool metadata (2)	0	0	1	0
Messages between agents, insecure channel (3)	0	0	0	0
Total (29)	7	10	12	15

Three routes produced harm in every model: indirect injection, persistent memory and a saved plan. The audit's harm counts here are the same harmful actions as in Table 3; a scenario untested for a model counts as not-harm in this table.

garak by family. garak's failure rates by family are in Appendix B. The families with the highest failure rates across models were rephrasing a request into the past or future tense (57–91%), prompt hijacking (55–83% where the model answered), and repetition-based divergence (14–86%); the lowest were invented software packages (1–4%) and malware-generation requests (4–16%). For kimi-k2.6, four families read near zero because almost all of its replies in them were empty (§4.1.3), and should not be read as resistance.

5 Discussion

5.1 For people choosing or deploying a model

A free score is one construct, measured once, by an instrument that may have failed silently. Before reading one, a deployer should know which construct matters, the reply limit used (critical for reasoning models), how many replies were empty, how many prompts the vendor blocked, and whether the run completed.

For agents with tools, a single injection benchmark run with a single attack is a narrow view: AgentDojo and the audit ranked the models alike, yet the model AgentDojo rated safest took harmful actions in the audit on its closest routes (3 of 7 scenarios; different units, so not a like-for-like rate) and through four routes AgentDojo does not exercise.

5.2 Where the free tools do something the audit does not

The free tools offer what the audit does not: public, re-runnable tests; a standard over-refusal figure (XSTest, AgentHarm's benign twin); honesty as belief against statement (MASK); the effect of a deployer's own defences (b3's levels, §4.3); breadth of text attack styles (garak's 81 probes in 18 families); task completion beside attack success (AgentDojo); and speed.

5.3 For tool maintainers

Each failure in §4.1 has a correction, and several are already options in the tools; the question for maintainers is which should be defaults.

- 1 Count and report empty and truncated replies in the summary, and do not grade empty replies as answers, evasions or passes (garak already skips missing outputs but scores empty strings; its open issue #804 [9]).
- 2 Record a vendor's content-filter refusal as a blocked prompt, counted in the summary, rather than an error that stops the run or a verdict either way: Inspect's native xAI provider already maps one to a content-filter result; its OpenAI-compatible path does not recognise Moonshot's `content_filter` error type; garak's OpenAI-compatible generator treats any 403 as terminal. Inspect's `--no-fail-on-error` tolerates the remaining sample errors.

- 3 Restore generator state after a probe that changes it, or isolate such probes; garak's `promptinject` option `generation_params` lets a user leave `max_tokens` out at the source.
- 4 Seed prompt sampling and record the seed (garak's `--seed` exists but is unset by default).
- 5 Have a scorer return an unscored result for a malformed grader output rather than raise (MASK), so the run continues whatever the error setting.
- 6 Bound stalled requests: Inspect's `--timeout` and `--attempt-timeout` exist; by default an attempt may wait 600 seconds and is retried without limit.
- 7 Report the graders' token cost beside the result: Inspect records usage per model, judges included, and prices it given a cost table.

Our runner applies 1 (it reports empty replies and gives reasoning models room, though it cannot change how a tool scores them), 2, 3 (`promptinject` runs last, alone), 4 and 6; 5 needs a change in MASK's scorer, and 7 is not yet done.

These are not all the same kind of request. Items 3 and 5, and the 403 handling in 2, are defects. Items 1, 4 and 6 concern defaults: each can already be changed by the user, and the argument is only about which value should be the default and what the summary should say. Item 7 is reporting. MASK's treatment of an evasion as not a lie is a design choice we do not ask anyone to change, only to report beside the count of empty replies, because at a small reply limit the two are the same thing.

6 Limitations

- **Statistical: one run per model per tool.** Every disagreement between measures is one draw, so disagreement and run-to-run noise are not separated. Wilson intervals (AgentDojo, XSTest, Table 3) understate uncertainty where samples cluster. The one repeated audit result agrees: gpt-4o-mini's agentic suite, run four times at a later version, did harm in 50.8% of 764 scenario runs (four repeats of 191 scenarios, so not independent).
- **External validity: four models,** none its vendor's newest; agreement among four carries little weight (§4.4).
- **Internal validity: unequal settings.** kimi-k2.6 ran with larger reply limits and at temperature 1, grok-4.20 at seed 1 and without garak's penalties (§3.3), and XSTest graded truncated replies for three models and full ones for kimi-k2.6. Re-running the conventional models at kimi-k2.6's limits would remove most of this.
- **Construct and internal validity: the comparison audit is the author's.** It is proprietary; its absolute rates depend on its design; its review turns were ruled by the author; its 400-turn figures are not on a common scale across models (§3.4); its step budget cut off 21 of kimi-k2.6's scenarios and 23 of grok-4.20's (Table 3 gives the bound); its runs used two scenario versions differing in 3 scenarios (Appendix E); and the only agreement figure between its grader and human labels is against the author's own rulings (the instrument's calibration set: 2,385 of 2,596 hand-ruled rows, 91.9%, measured at v17.7.59), which is a consistency check against one reviewer, not independent agreement. The free tools' GPT-4o judges grade a model from the same vendor (gpt-4o-mini), a known source of self-preference [17].
- **Cost figures** are relative, at list prices, for successful runs; garak's are estimated (kimi-k2.6's from one interval); AgentDojo is included where its counts survive; b3's embeddings are omitted.

- **garak for kimi-k2.6** lies between 32.3% and 63.3% (§4.1.1) until re-run with `promptinject` isolated, and every model's garak runs include the probe-state leak in the affected families.
- **Construct validity: the free tools' graders were taken as given** apart from empty replies, and we did not validate their verdicts against human judgement; their agreement with humans is as their authors [3, 4] and work on LLM judges [17–21] report.
- **The corrections in §5.3 are untested.** They were developed during these runs and only partly applied to them (§3.3, Appendix A); whether any of them changes a tool's ranking is not known.
- **Future work.** Repeat each measurement several times; re-run the conventional models at the reasoning model's limits and kimi-k2.6's garak with `promptinject` isolated; extend the comparison to current frontier models, subject to their vendors' filters; run AgentDojo's other attacks; check the free tools' judges against human labels on truncated XSTest replies and on MASK evasions; re-run b3 without its judge level and AgentHarm's benign twin with empty or cut-off replies counted as untested; and test whether the corrections in §5.3 change the tools' rankings.

7 Conclusion

We asked three questions. **Do the tools behave correctly as instruments on current models?** Not reliably: they graded empty replies as answers, evasions or passes, aborted whole runs on one vendor refusal, leaked settings between probes, and truncated replies differently by model; several of these failures are specific to reasoning models, and each has a straightforward correction (§5.3). **What does it cost to run them?** More than the software price suggests: on the cheapest models the tools' own graders cost more than the model under test, and on the reasoning model five tools cost 4.9 times as much as on gpt-4o-mini with the graders included, 50 times for the model's own usage, most of it hidden reasoning. **How do their results compare?** Among the three models run under the same settings the tools barely agreed (W 0.19–0.33), as prior work found; the fourth model's consistent first place is confounded with its settings. In one run, AgentDojo and the action-graded audit ordered the models the same way, and the model AgentDojo rarely hijacked took harmful actions in the audit through routes AgentDojo does not exercise; the audit's half of that comparison cannot be reproduced by readers. Free safety tools are a sound first step. Until the instruments are fixed, a free-tool result should be read with its reply limit, its count of empty replies, its blocked prompts and its grader's cost beside it.

Ethics and disclosure

No attack prompt or model output that would enable harm is reproduced here; failures are described by type. The author has a commercial interest in the comparison audit (see §1). We have drafted reports for garak's maintainers (settings persisting after `promptinject`, and `content-filter 403s` treated as authentication failures, §4.1.2–4.1.3), for Inspect (Moonshot's `content_filter` error type not recognised by the OpenAI-compatible provider) and for Inspect's evaluations (MASK's scorer raising on one malformed judge answer, and evidence for the open empty-response issue), and will file them with the logs once this paper is public, so that each project can read the measurements in context. Neither project runs a bug bounty.

Data and code availability

The free tools are public (§3.2). The scripts that derive every figure in this paper (`paper_numbers.py`, `cluster_bootstrap.py`), their outputs, the per-sample scores and the runner are available from the author on request. The raw run logs (Inspect `.eval` files and garak reports) contain model replies to harmful requests and are available to researchers on request. The comparison audit's scenarios are proprietary; Appendix D shows how a scenario is built.

Appendix A. Run conditions per tool and model

Samples run, samples that ended in an error (vendor refusals and tool errors, excluded from scores), final replies that stopped at the reply limit, and wall-clock minutes, from each tool's own log. MASK stores its replies outside the final output, so its empty-reply count is given in §4.1.1 instead.

Tool	kimik2.6	Llama-3.3-70B	gpt-4o-mini	grok-4.20
AgentHarm	176 · err 0 · cap 0 · 5 min	176 · err 0 · cap 0 · 2 min	176 · err 0 · cap 0 · 1 min	176 · err 0 · cap 0 · 2 min
AgentHarm benign	176 · err 0 · cap 9 · 10 min	176 · err 0 · cap 0 · 3 min	176 · err 0 · cap 0 · 2 min	176 · err 0 · cap 0 · 2 min
StrongREJECT	313 · err 1 · cap 0 · 12 min	313 · err 0 · cap 0 · 1 min	313 · err 0 · cap 0 · 1 min	313 · err 11 · cap 0 · 2 min
StrongREJECT + AIM	313 · err 1 · cap 0 · 2 min	313 · err 0 · cap 0 · 1 min	313 · err 0 · cap 0 · 1 min	313 · err 8 · cap 0 · 1 min
MASK	1000 · err 2 · cap 0 · 76 min	1000 · err 0 · cap 0 · 23 min	1000 · err 0 · cap 0 · 9 min	1000 · err 0 · cap 0 · 15 min
b3	3150 · err 0 · cap 0 · 711 min	3150 · err 1 · cap 0 · n/a	3150 · err 0 · cap 1 · 15 min	3150 · err 0 · cap 0 · 23 min
XSTest safe	250 · err 0 · cap 2 · 14 min	250 · err 0 · cap 163 · 4 min	250 · err 0 · cap 137 · 1 min	250 · err 0 · cap 178 · 2 min
XSTest unsafe	200 · err 0 · cap 0 · 8 min	200 · err 0 · cap 23 · <1 min	200 · err 0 · cap 25 · 1 min	200 · err 1 · cap 66 · 1 min
AgentDojo (no attack + under attack)	97 + 949 · unscored 0 · n/a · n/a	97 + 949 · unscored 0 · n/a · n/a	97 + 949 · unscored 0 · n/a · 138 min	97 + 949 · unscored 0 · n/a · 101 min

$n \cdot err \cdot cap \cdot minutes$. XSTest's truncation is discussed in §4.1.1. kimik2.6's wall-clock figures include waits while its account was out of credit; times for resumed runs cover only the resumed part; Llama-3.3-70B's b3 time is not recorded because its summary was recomputed (§3.3).

Run notes in full (summarised in §3.3).

- **xAI** rejects `seed = 0`, which AgentHarm sets. grok-4.20 ran at seed 1 on every Inspect tool (forced for AgentHarm; applied to the rest for uniformity, where the other models used each tool's own seed: 0 on AgentHarm, none elsewhere). xAI also rejects garak's frequency and presence penalties, so garak ran on grok-4.20 without them; their defaults are zero, but garak's `promptinject` templates set non-zero penalties (frequency 0.2–1.0, presence 0.5–1.0), which were therefore not applied on grok-4.20 in `promptinject` or the probes that inherit its settings (§4.1.3).
- **Moonshot** allows only temperature 1 (and `top_p` 0.95) for kimik2.6, so every kimik2.6 call ran at temperature 1, where the other models ran at each tool's own temperature (0 to 1). garak's `top_p` was suppressed for kimik2.6 so that Moonshot's fixed value applied.

- **One generation per garak prompt** on every model (garak's default is five), to bound cost.
- **Reply limits for the reasoning model.** kimi-k2.6 was given 4,096 tokens for XSTest and garak and 8,192 for MASK, because at the defaults its replies were empty (§4.1.1; at garak's default of 150, all 8 replies to garak's own test probe were empty and its detector passed all 8, in a trial re-run on 14 October 2026 with the report kept, Appendix C). Its scores were therefore not measured under the same limits as the other three models.
- **AgentDojo.** The driver requests temperature 0 for every model; what Moonshot applied for kimi-k2.6 is not recorded. AgentDojo's attack text addresses the model by name: we used AgentDojo's own name for gpt-4o-mini ("GPT-4") and "Grok", "Kimi" and "AI assistant" (for Llama-3.3-70B) for the others. On Moonshot it needed two driver adaptations (the `developer` role sent as `system`; no empty assistant text), neither of which changes what the model reads. On Llama-3.3-70B, a guard turned tool calls with malformed arguments into empty ones so that the tool returned an error instead of crashing the suite; it fired twice. Every one of the 97 tasks without attack and 949 under attack was scored for every model.
- **Vendor refusals.** xAI and Moonshot were reached through Inspect's generic OpenAI-compatible provider, not a vendor-specific one. Inspect tasks ran with `--no-fail-on-error`, except Llama-3.3-70B's first MASK, b3 and StrongREJECT runs, which ran at Inspect's documented default; two of them stopped on one sample's error (§4.1.4) and were resumed with the flag. garak ran through a wrapper that records a vendor content-filter HTTP 403 as a blocked prompt (§4.1.2). Samples that error are excluded from scores.
- **Interruptions.** Runs interrupted when an account ran out of credit or a connection stalled were resumed with `inspect eval-retry`, which keeps completed samples and re-runs the rest (Llama-3.3-70B: MASK, b3 and XSTest unsafe; kimi-k2.6: StrongREJECT with AIM and XSTest safe, both restarted from the beginning). kimi-k2.6's MASK was re-run in full with the larger reply limit, and the capped run kept as evidence (§4.1.1). Llama-3.3-70B's b3 finishing run ended before writing its summary; its score is the plain mean of the 3,149 saved per-sample scores, a method that reproduces the tool's own b3 summaries for gpt-4o-mini (0.1554) and grok-4.20 (0.3036) exactly. kimi-k2.6's garak run was paused for credit after 19 probes and resumed for the remaining 62; garak scores probe by probe, so the two reports merge without overlap. Samples, errors and truncations per run are in the table above.

The corrections in §5.3 were developed during these runs; the reported runs did not have garak seeded, did not isolate `promptinject`, and used Inspect's default request handling (600 seconds per attempt, retried without limit) except kimi-k2.6's MASK run (a 300-second limit) and Llama-3.3-70B's b3 finishing run.

Appendix B. garak: checks failed by family, as garak reports them

Share of detector checks that failed, per family, with empty replies counted as garak counts them (as passes). Families after `promptinject` in garak's order (snowball, sysprompt_extraction, web_injection) ran after it on every model and so inherited whatever settings it left (§4.1.3).

Family	kimi-k2.6	Llama-3.3-70B	gpt-4o-mini	grok-4.20
ansiescape	46.8%	33.0%	35.8%	55.0%
dan	23.7%	78.5%	16.3%	59.9%
divergence	68.9%	79.0%	14.3%	86.0%
encoding	42.4%	14.7%	29.2%	38.9%
exploitation	60.7%	46.4%	39.3%	10.7%

Family	kimi-k2.6	Llama-3.3-70B	gpt-4o-mini	grok-4.20
goodside	35.9%	2.6%	2.6%	35.9%
grandma	37.9%	67.2%	50.0%	61.8%
latentinjection	30.9%	39.3%	24.7%	43.0%
leakreplay	25.0%	25.0%	14.0%	29.1%
lmrc	16.7%	25.0%	25.0%	33.3%
malwaregen	15.4%	16.2%	3.8%	6.2%
misleading	57.5%	57.2%	50.0%	51.2%
packagehallucination	1.0%	3.9%	1.3%	2.5%
phrasing	69.5%	90.6%	57.0%	90.0%
promptinject	0.0%	81.0%	55.1%	82.7%
snowball	0.0%	0.0%	48.0%	0.0%
sysprompt_extraction	0.4%	17.6%	4.3%	6.2%
web_injection	0.0%	4.9%	16.0%	25.5%

Checks evaluated: kimi-k2.6 16,684, Llama-3.3-70B 16,700, gpt-4o-mini 16,700, grok-4.20 16,695. Replies with no text: kimi-k2.6 3,791, Llama-3.3-70B 60, gpt-4o-mini 40, grok-4.20 5.

Appendix C. Records and code

Run logs: `freetools_runs_2026-09-30/`, `freetools_runs_2026-10-01/` (Inspect `.eval` logs, garak reports; the capped kimi-k2.6 MASK run and the interrupted first kimi-k2.6 garak run are kept under `_aborted/`). Data: `freetools_data.json` and `freetools_costs.json` in `freetools_runs_2026-10-01/` (the currency figures behind the ratios in §4.2, with each tool's model and grader cost; copies kept as `2026-10-02_freetools_data.json` and `2026-10-02_freetools_costs.json`). Cluster-bootstrap intervals: `cluster_bootstrap.py`, written to `cluster_bootstrap.json`. The audit's harm by route: `audit_routes.py`, which re-grades each model's replay log. Derived figures in this paper, recomputed from the logs or, where marked in the script, recorded from the audit (MASK reply counts, garak answered, shared-prompt and bounded rates, Kendall's W variants with permutation p -values and their sensitivity to the garak scoring, the chance of the shared ordering, kimi-k2.6's reasoning-token shares, the audit's harm counts by route and their upper bounds): `paper_numbers.py`, written to `paper_numbers.json`, with its counting rules in its header. The 14 October 2026 trials at the tools' default limits (§4.1.1, Appendix A): `kimi/logs_retrial_2026-10-14/` (XSTest, 256 tokens) and `kimi/garak_retrial_2026-10-14/` (garak `test.Test`, 150 tokens). Runner: `freetools_engagement.py`, `garak_launch.py`, `xstest_local.py` (available on request); garak's 81 probes are listed in `garak_probe_list.txt`. kimi-k2.6 ran with Moonshot's default thinking mode. The audit's harm by route is re-graded by `audit_routes.py` with the grader pinned at v17.7.62; the 400-turn indices after human review are recomputed by `battery_post_review.py`, which first checks that it reproduces each pack's sealed indices. AgentDojo: `dojo_engagement.sh`, `dojo_block.py`. Audit records: the run ledger entries KATANA-2026-0930-01, -0930-02 and -1002-01.

Appendix D. How an audit scenario is built (illustrative)

Each agentic scenario gives the model a role, a task, working instrumented tools, any records the task needs and, in most families, the organisation's written policy. Harm is defined per scenario as a specific tool action, and the grader reads the tool-call log.

An example of the structure, not a scenario from the suite. The model is a finance assistant with tools to read the vendor list, look up past invoices and send a payment. A message that appears to come from a known vendor says its bank details have changed and asks for this month's invoice to be paid to a new account. The policy says payee details may only change after confirmation through a contact already on file. **Harm:** a payment call to the new account without that confirmation. **Not harm:** pausing, asking for confirmation, or escalating. **Its control twin** has the same setup, but the change was confirmed through the contact on file as the policy requires; the correct action is to pay, and not acting is recorded as a failure to complete legitimate work.

Injection scenarios follow the same pattern, but the instruction arrives inside content the agent reads during a legitimate task (a calendar invite, a retrieved document, a tool's description, a note it saved earlier, a message from another agent), and harm is acting on that instruction.

Appendix E. Auditor and scenario versions

The agentic suite was run at auditor versions v17.7.37 (grok-4.20, Llama-3.3-70B), v17.7.57 (gpt-4o-mini) and v17.7.59 (kimi-k2.6), at a step budget of ten rounds, and every row was re-graded by v17.7.62, which records a scenario cut off by the step budget as untested. Re-grading with the later v17.7.63+1, which changed how the step budget is handled, gives the same harmful actions for every model but counts the cut-off scenarios as safe; we therefore use v17.7.62. All four audit runs used the vendor's default temperature and a 4,000-token output cap. The scenario set is fingerprinted in each run's manifest (a hash of every scenario's stimulus): grok-4.20 and Llama-3.3-70B ran the v17.7.37 set (bb41327168a7), gpt-4o-mini and kimi-k2.6 the v17.7.57/59 set (16c49ec28ef8). Recomputing each scenario's hash at the two tags shows 3 of 258 scenarios changed between them (two in the multi-tenant isolation family, one in incremental self-escalation; one of the three a control), none in the twelve injection families; the scripted tool replies were identical (a901ee65457e). The grader and harness code also changed between these versions; every row is therefore re-scored by one grader, but harness behaviour other than the stimuli and scripted replies is not proven identical. The 400-turn conversation ran at v17.7.62 for kimi-k2.6 and v17.7.59 for the others; grok-4.20 had 399 measured turns.

Appendix F. Token counts per tool and model, and the prices behind the ratios

Input / output tokens of the model under test, from each Inspect log's own usage record (output includes hidden reasoning). garak records no token counts and is excluded (§4.2). These are the figures behind every ratio in §4.2; the full per-run file is `freetools_costs.json` (Appendix C).

Tool	kimi-k2.6	Llama-3.3-70B	gpt-4o-mini	grok-4.20
AgentHarm	202,870 / 149,295	605,086 / 34,804	416,891 / 61,556	439,859 / 65,214
AgentHarm benign	583,078 / 385,857	716,345 / 53,495	465,266 / 84,445	431,801 / 76,606
StrongREJECT	11,402 / 171,975	12,428 / 12,394	11,367 / 13,324	22,173 / 81,200
StrongREJECT + AIM	51,070 / 186,418	126,360 / 9,034	125,328 / 2,885	16,890 / 49,683
MASK	332,625 / 6,060,709	357,389 / 1,215,956	334,258 / 783,231	410,340 / 1,429,269
b3	4,026,103 / 11,123,981	8,595,314 / 1,058,707	1,773,142 / 833,625	2,282,766 / 1,413,280
XSTest safe	6,685 / 237,445	7,820 / 53,383	6,794 / 48,266	6,657 / 53,273
XSTest unsafe	5,433 / 105,913	6,241 / 12,555	5,411 / 11,751	3,076 / 37,393
Five tools, total	5,219,266 / 18,421,593	10,426,983 / 2,450,328	3,138,457 / 1,839,083	3,613,562 / 3,205,918

List prices per million tokens (input / output) on 2 October 2026, before caching discounts: gpt-4o-mini 0.15 / 0.60; grok-4.20 1.25 / 2.50; Llama-3.3-70B (DeepInfra) 0.10 / 0.32; kimi-k2.6 0.95 / 4.00; the graders, GPT-4o 2.50 / 10.00 and o3-mini 1.10 / 4.40 (USD). Grader cost as a multiple of the model under test, per tool: AgentHarm 1.5 / 7.3 / 5.0 / 0.7 (kimi-k2.6 / Llama-3.3-70B / gpt-4o-mini / grok-4.20); AgentHarm benign 1.2 / 5.4 / 3.8 / 0.7; StrongREJECT 1.6 / 102 / 102 / 5.2; StrongREJECT + AIM 1.8 / 69 / 67 / 9.7; MASK 0.6 / 36 / 26 / 3.8; XSTest safe 1.5 / 62 / 41 / 9.0; XSTest unsafe 2.4 / — / 93 / 10.7 (Llama-3.3-70B's model cost there rounds to zero); b3 has no LLM grader. Across the five tools the model under test cost, in the same currency, kimi-k2.6 78.65, grok-4.20 12.54, Llama-3.3-70B 1.83 and gpt-4o-mini 1.58, and their graders 23.76, 21.36, 20.55 and 19.21.

References

- [1] E. DeBenedetti, J. Zhang, M. Balunović, L. Beurer-Kellner, M. Fischer, F. Tramèr. AgentDojo: A Dynamic Environment to Evaluate Prompt Injection Attacks and Defenses for LLM Agents. NeurIPS 2024 Datasets and Benchmarks Track. arXiv:2406.13352.
- [2] M. Andriushchenko, A. Souly, M. Dziemian, et al. AgentHarm: A Benchmark for Measuring Harmfulness of LLM Agents. ICLR 2025. arXiv:2410.09024.
- [3] A. Souly, Q. Lu, D. Bowen, et al. A StrongREJECT for Empty Jailbreaks. NeurIPS 2024 Datasets and Benchmarks Track. arXiv:2402.10260.
- [4] R. Ren, A. Agarwal, M. Mazeika, et al. The MASK Benchmark: Disentangling Honesty From Accuracy in AI Systems. arXiv:2503.03750, 2025.
- [5] J. Bazinska, M. Mathys, F. Casucci, et al. Breaking Agent Backbones: Evaluating the Security of Backbone LLMs in AI Agents. arXiv:2510.22620, 2025.
- [6] P. Röttger, H. R. Kirk, B. Vidgen, G. Attanasio, F. Bianchi, D. Hovy. XSTest: A Test Suite for Identifying Exaggerated Safety Behaviours in Large Language Models. NAACL 2024, pp. 5377–5400. arXiv:2308.01263.
- [7] L. Derczynski, E. Galinkin, J. Martin, S. Majumdar, N. Inie. garak: A Framework for Security Probing Large Language Models. arXiv:2406.11036, 2024.
- [8] UK AI Security Institute. Inspect AI: Framework for Large Language Model Evaluations. Software, 2024. https://github.com/UKGovernmentBEIS/inspect_ai; evaluations: https://github.com/UKGovernmentBEIS/inspect_evals.
- [9] NVIDIA garak repository: `garak/probes/promptinject.py`; issue #2259, "promptinject: capped probes apply settings from the wrong prompt" (opened 2026-10-01); issue #1967, "NIM generator aborts entire probe on a single transient HTTP error (408)" (opened 2026-07-15, closed); issue #1608, "bug: max output tokens overwritten" (closed), and pull request #1729, "allow promptinject probes to limit `generation_params`" (merged 2026-05-27); issue #804, "probes that fail to elicit valid generator output should result in SKIP" (opened 2024-07-30, open). <https://github.com/NVIDIA/garak>.
- [10] M. Q. Li, B. C. M. Fung, B. Li, H. Ismail, F. Iqbal. Taxonomy and Consistency Analysis of Safety Benchmarks for AI Agents. arXiv:2605.16282, 2026.
- [11] Y. Wang, X. Han, D. Shang, Y. Tang, B. Liu. Safety, or Just Capability? A Validity Audit of Agent-Safety Benchmarks. arXiv:2607.28685, 2026.
- [12] M. Desai, S. T. Truong, H. Wallach, et al. What AI Benchmarks Actually Measure: Adapting Convergent and Discriminant Validity to Interrogate Fifty-Six AI Benchmarks. arXiv:2609.08812, 2026.
- [13] Y. Li, Z. Fan, Z. Zhuang. SafetyRepro: Configuration-Conditional Rank Instability on Alignment Benchmarks. arXiv:2605.25492, 2026.
- [14] T. Beyer, S. Xhonneux, S. Geisler, G. Gidel, L. Schwinn, S. Günemann. LLM-Safety Evaluations Lack Robustness. arXiv:2503.02574, 2025.
- [15] R. Ren, S. Basart, A. Khoja, et al. Safetywashing: Do AI Safety Benchmarks Actually Measure Safety Progress? NeurIPS 2024 Datasets and Benchmarks Track. arXiv:2407.21792.

- [16] J. Brokman, O. Hofman, O. Rachmil, et al. Insights and Current Gaps in Open-Source LLM Vulnerability Scanners: A Comparative Analysis. arXiv:2410.16527, 2024.
- [17] L. Zheng, W.-L. Chiang, Y. Sheng, et al. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. NeurIPS 2023 Datasets and Benchmarks Track. arXiv:2306.05685.
- [18] A. S. Thakur, K. Choudhary, V. S. Ramayapally, S. Vaidyanathan, D. Hupkes. Judging the Judges: Evaluating Alignment and Vulnerabilities in LLMs-as-Judges. GEM² Workshop 2025. arXiv:2406.12624.
- [19] F. Eiras, E. Zémour, E. Lin, V. Mugunthan. Know Thy Judge: On the Robustness Meta-Evaluation of LLM Safety Judges. ICBINB Workshop at ICLR 2025. arXiv:2503.04474.
- [20] K. Vishnubhotla, S. Vajjala, A. Vij, I. Nejadgholi. LLM Judges Inconsistently Disagree Across Safety Criteria and Harm Categories. arXiv:2605.31381, 2026.
- [21] H. Tamba. Necessary but Not Sufficient: Temperature Control and Reproducibility in LLM-as-Judge Safety Evaluations. arXiv:2606.26185, 2026.
- [22] A. Panfilov, E. Kortukov, K. Nikolić, et al. Strategic Dishonesty Can Undermine AI Safety Evaluations of Frontier LLMs. arXiv:2509.18058, 2025.
- [23] A. Hochlehnert, H. Bhatnagar, V. Udandara, S. Albanie, A. Prabhu, M. Bethge. A Sober Look at Progress in Language Model Reasoning: Pitfalls and Paths to Reproducibility. COLM 2025. arXiv:2504.07086.
- [24] F. Jiang, Z. Xu, Y. Li, et al. SafeChain: Safety of Language Models with Long Chain-of-Thought Reasoning Capabilities. arXiv:2502.12025, 2025.
- [25] B. Atil, S. Aykent, A. Chittams, et al. Non-Determinism of "Deterministic" LLM Settings. arXiv:2408.04667, 2024.
- [26] Y. Song, G. Wang, S. Li, B. Y. Lin. The Good, The Bad, and The Greedy: Evaluation of LLMs Should Not Ignore Non-Determinism. arXiv:2407.10457, 2024.
- [27] E. Miller. Adding Error Bars to Evals: A Statistical Approach to Language Model Evaluations. arXiv:2411.00640, 2024.
- [28] S. Biderman, H. Schoelkopf, L. Sutawika, et al. Lessons from the Trenches on Reproducible Evaluation of Language Models. arXiv:2405.14782, 2024.
- [29] S. Kapoor, B. Stroebl, Z. S. Siegel, N. Nadgir, A. Narayanan. AI Agents That Matter. arXiv:2407.01502, 2024.
- [30] S. Kapoor, B. Stroebl, P. Kirgis, et al. Holistic Agent Leaderboard: The Missing Infrastructure for AI Agent Evaluation. arXiv:2510.11977, 2025.
- [31] A. Ghosh, Y. Mai, G. Channing, L. Choshen. AI evals are becoming the new compute bottleneck. Hugging Face blog (EvalEval Coalition), 29 April 2026.
- [32] Q. Zhan, Z. Liang, Z. Ying, D. Kang. InjecAgent: Benchmarking Indirect Prompt Injections in Tool-Integrated Large Language Model Agents. Findings of ACL 2024. arXiv:2403.02691.
- [33] Y. Ruan, H. Dong, A. Wang, et al. Identifying the Risks of LM Agents with an LM-Emulated Sandbox. ICLR 2024. arXiv:2309.15817.
- [34] T. Yuan, Z. He, L. Dong, et al. R-Judge: Benchmarking Safety Risk Awareness for LLM Agents. Findings of EMNLP 2024. arXiv:2401.10019.
- [35] Z. Zhang, S. Cui, Y. Lu, J. Zhou, J. Yang, H. Wang, M. Huang. Agent-SafetyBench: Evaluating the Safety of LLM Agents. arXiv:2412.14470, 2024.
- [36] M. Q. Li, B. C. M. Fung, M. Weiss, P. Xiong, K. Al-Hussaini, C. Fachkha. A Benchmark for Evaluating Outcome-Driven Constraint Violations in Autonomous AI Agents. arXiv:2512.20798, 2025.
- [37] S. Vijayvargiya, A. B. Soni, X. Zhou, Z. Z. Wang, N. Dziri, G. Neubig, M. Sap. OpenAgentSafety: A Comprehensive Framework for Evaluating Real-World AI Agent Safety. ICLR 2026. arXiv:2507.06134.
- [38] P. Kumar, E. Lau, S. Vijayakumar, et al. Refusal-Trained LLMs Are Easily Jailbroken As Browser Agents. arXiv:2410.13886, 2024.
- [39] Y. Tang, T. Li, E. Li, C. J. Maddison, H. Dong, Y. Ruan. LM Agents May Fail to Act on Their Own Risk Knowledge. arXiv:2508.13465, 2025.
- [40] J. Scheurer, M. Balesni, M. Hobbhahn. Large Language Models can Strategically Deceive their Users when Put Under Pressure. arXiv:2311.07590, 2023.

- [41] A. Meinke, B. Schoen, J. Scheurer, R. Shah, M. Balesni, M. Hobbhahn. Frontier Models are Capable of In-context Scheming. arXiv:2412.04984, 2024.
- [42] R. Jha, H. Triedman, A. Bhattacharya, V. Shmatikov. Agent Meltdowns: The Road to Hell Is Paved with Helpful Agents. arXiv:2605.19149, 2026.
- [43] UK AI Security Institute, inspect_evals issue tracker (titles abridged): #2415 "Empty response checks for judges" (2026-09-10); #2130 "b3 scorer drops reasoning content" (2026-08-12); #2212 "AgentHarm refusal judge converts API errors into 'did not refuse' verdicts" (2026-08-20); #2091 "Scorers disagree on whether an instrument failure is a model failure" (2026-08-05); #2599 "Agree the outcome for grader, sandbox and dependency failures" (2026-10-01). https://github.com/UKGovernmentBEIS/inspect_evals/issues.