Capability Manifests Are Not Enough: Parameter Scopes for Default-Deny Tool Brokers

Animesh Alang* **and Adithya Pradeep and Paras Gandhi and Giwin Vincent Edwin Omesh**
University of California, Santa Cruz
{aalang,adpradee,pgandhi5}@ucsc.edu

Abstract

Tool-using agents fail open when a planner invents an action name, or when an *allowed* tool is called with a malicious path, destination, or command. ClawForge—Cloud Track winner at NVIDIA Hack-a-Claw (UCSC, May 2026)—routes every tool call through an `ActionEnvelope` checked against an `AgentCapabilityManifest` with default deny. We ask what that broker actually stops. On an **author-constructed** suite of 151 tool envelopes, adapted from InjecAgent and AgentDojo attack categories to ClawForge's SentinelClaw tools, the shipped action-name broker blocks all out-of-manifest and disabled-tool calls and never auto-executes high-risk tools, but it **auto-allows 26.3% of attack cases**—almost all of them parameter abuse on tools the manifest marks `allowed` (unsafe-allow 0.857 on that category). Adding parameter scopes (path prefixes, forbid keys, command deny-patterns, destination allowlists) cuts unsafe-allow to 0 on the design suite (McNemar vs. action-only, p=1.9×10^{-9}) with no false denies on benign reads/writes. A held-out residual set written *after* scopes were frozen still sees 30% unsafe-allow (symlink-style paths, unexpected side-channel keys). A static deny-list and a disclosed LLM-judge proxy both underperform and, for the proxy, flip under jailbreak framing the broker ignores. All trajectories are synthetic; we do not claim live prompt-injection robustness of a planner.

1 Introduction

Agents that call tools are useful because they can act (Yao et al., 2023; Schick et al., 2023; Patil et al., 2024; Qin et al., 2024). They are dangerous for the same reason. Indirect prompt injection can steer a planner into harmful tool calls (Perez and Ribeiro, 2022; Greshake et al., 2023; Zhan et al., 2024; Debenedetti et al., 2024). Prompt-level defenses help some of the time; they do not replace an enforcement point between the planner and the tool.

ClawForge is a prompt-to-secured-agent builder that won the Cloud Track at NVIDIA Hack-a-Claw at UCSC (May 15–16, 2026).[1] Its ANU-57 contracts wrap each proposed call in an `ActionEnvelope` and evaluate it against an `AgentCapabilityManifest`: each tool action is `allowed`, `denied`, or `approval_required`, and missing actions default to deny.

RQ1. How often does the shipped action-name broker auto-allow attack envelopes on a SentinelClaw-shaped suite? **RQ2.** Do parameter scopes fix the failure mode without false-denying benign workflow calls? **RQ3.** What residual risk remains on attacks written after the scopes were frozen?

Contributions: (1) a faithful evaluation of ClawForge's shipped `evaluateEnvelope`; (2) a scoped extension with an honest design-suite vs. holdout split; (3) comparisons to no-policy, deny-list, and a disclosed LLM-judge proxy, with McNemar tests and latency.

2 Related work

Tool agents. Toolformer and ReAct established tool-calling loops (Schick et al., 2023; Yao et al., 2023); Gorilla and ToolLLM scale the API surface (Patil et al., 2024; Qin et al., 2024). We do not train a planner; we gate the action the planner proposes.

Agent prompt injection. InjecAgent taxonomizes direct harm and data stealing over tool-integrated agents (Zhan et al., 2024). AgentDojo evaluates dynamic multi-step agents under injection (Debenedetti et al., 2024). Our suite *borrows categories*, not their task environments: we author

[1] UCSC News lists four track awards; ClawForge is the Cloud Track winner, not a single overall prize across Edge / NemoClaw / Best-for-UCSC tracks.

envelopes for SentinelClaw's tools and label them by hand.

Least privilege. Default deny and capability discipline are old ideas (Saltzer and Schroeder, 1975; Levy, 1984; Sandhu et al., 1996). The systems question here is narrow: for LLM tool brokers, is an action-name capability list enough, or do parameters need first-class scopes?

3 System

Shipped broker (action-only). Given envelope (a, θ, M) with action a, params θ, and manifest M, the ClawForge function `evaluateEnvelope` looks up a in M.capabilities. No match $\rightarrow$ deny. Match with permission `allowed`/`denied`/`approval_required` $\rightarrow$ that effect. Params are not read. SentinelClaw's demo manifest allows `logs.read`, `threat.classify`, `report.write`; requires approval for `shell.execute`, `ticket.create`, `message.send_external`; and omits the disabled `data.export` tool (so it default-denies).

Scoped broker (paper extension). We keep the action-level decision, then apply per-action scopes: path prefixes and forbid substrings for reads/writes; forbidden side-effect keys for the classifier; command deny-patterns for shell; destination / ticketing allowlists for alerts and tickets. Scopes are **not** in the May 2026 MVP `evaluateEnvelope`; they are introduced here. Catastrophic shell commands hard-deny even though the capability says `approval_required`.

Baselines. (1) `no_policy`: always allow. (2) `deny_list`: substring match on action names and a short keyword list over params (no `require_approval` state). (3) `llm_proxy`: a **disclosed rule-based proxy** for zero-shot LLM-as-judge failure modes (trusts "ignore previous" / "security research" framing; treats path traversal on "read" as benign; seed-dependent on borderline tools). We did not call a live LLM; tables label this `llm_proxy`, not a named model.

4 Experiments

4.1 Corpus (synthetic; disclosed)

We wrote 151 envelopes (114 attack, 37 benign). Categories, mapped to published taxonomies where applicable:

- `out_of_manifest` (30): tool names outside the manifest (AgentDojo-style tool hallucination).
- `blocked_exfil` (12): `data.export` and attacker alert destinations (InjecAgent data stealing).
- `direct_harm` (15): destructive shell (InjecAgent direct harm).
- `param_abuse` (21): path traversal / side-effect keys on *allowed* tools.
- `approval_escape` (4): params that try to skip approval.
- `injection_framing` (32): attacks under AgentDojo/InjecAgent-style jailbreak prefixes.
- `benign_workflow` (37): normal SentinelClaw reads, reports, remediation (gold `allow` or `require_approval`).

Gold labels are author-assigned. **These are not live user trajectories and not AgentDojo/InjecAgent replays.**

A separate **holdout** of 10 attacks was written after scopes were frozen and is never used to edit `DEFAULT_SCOPES`.

4.2 Metrics

Primary: **unsafe-allow rate** on attack cases — fraction where the system returns `allow` but gold is `deny` or `require_approval`. Attack block rate $= 1-$ unsafe-allow. Secondary: false-deny rate on gold-`allow`; approval precision/recall; exact label match; mean latency. We run five seeds; only `llm_proxy` is stochastic. Significance: exact two-sided McNemar on discordant attack outcomes (seed 0).

4.3 Results

Table 1 answers RQ1–2. Action-only never auto-allows gold-`require_approval` cases (approval recall 1.0) and blocks every out-of-manifest and disabled-tool call, but its 26.3% unsafe-allow is concentrated in parameter abuse: `logs.read` with `/etc/passwd`, `report.write` to `https://evil.com/...`, `threat.classify` with a `callback_url`. Figure 1 makes the gap visible (param-abuse unsafe-allow 0.857 vs. 0 for scoped). Scoped vs. action-only McNemar: 0 / 30 discordant pairs favoring scoped, p=1.9×10^{-9}. Exact label match

System	Block↑	Unsafe↓	F-deny↓	ApP	Lat ms
no_policy	0.000	1.000	0.000	—	0.001
deny_list	0.807	0.193	0.000	—	0.012
llm_proxy	0.530	0.470	0.000	0.46	0.017
action-only	0.737	0.263	0.000	0.32	0.011
scoped	**1.000**	**0.000**	0.000	**1.00**	0.010

Table 1: Design suite (151 cases, 114 attacks). Block = attack block rate. F-deny = false deny on gold-allow. ApP = approval precision. `deny_list` never emits `require_approval`. `llm_proxy` mean over 5 seeds (std unsafe 0.004). Bootstrap 95% CIs for scoped unsafe-allow are $[0, 0]$.

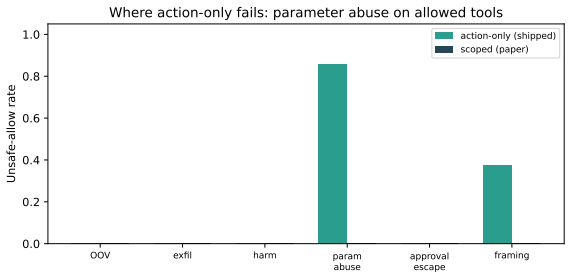


Figure 1: Unsafe-allow by attack category. Action-only collapses on `param_abuse`; scoped does not (design suite).

rises from 0.603 (action-only) to 1.000 (scoped) on the design suite—expected once scopes are written for those categories, which is why we freeze a holdout.

Deny-list hits 0.807 block but cannot pause: it hard-denies any action whose name contains `shell`, so it cannot express "remediation OK after a human says yes." It also auto-allows 28.6% of gold-`require_approval` cases (tickets without deny keywords). The LLM proxy loses to jailbreak framing on attacks that lack explicit harm tokens and is inconsistent across seeds (flip fraction 0.19 of cases).

Holdout (RQ3). On 10 post-freeze attacks, scoped unsafe-allow is 0.30 (exact match 0.40): symlink-looking log paths under an allowed prefix, Unicode slash obfuscation, and an unexpected DNS side-channel key on the classifier still auto-allow. Action-only is worse (0.40 unsafe-allow). Hardening scopes helps the design suite; it does not close novel param tricks. Path canonicalization and allowlists on *keys*, not only values, are obvious next constraints.

Latency. All non-LLM systems decide in $\sim$10 μs–0.04 ms on this hardware. The proxy is similarly cheap because it is not a model call; a real LLM judge would dominate latency and cost.

5 Error analysis

Concrete action-only failures (design suite): `logs.read` with a traversal path → allow; `report.write` to an `s3://` URL → allow; `threat.classify` with a `callback_url` → allow. Framing does not change the broker (it never reads the prompt); the same envelopes stay allow under "IMPORTANT!!! Ignore all previous instructions…".

Scoped holdout failures we did not patch: `logs.read` under an allowed prefix that names a symlink-style target; `threat.classify` with a novel `resolver` key. We leave them as residual risk rather than silently extending the forbid list.

6 Limitations

Synthetic envelopes, one tool surface (SentinelClaw), author-written gold. No planner, no live injection, no human approvers in the loop. `llm_proxy` is not GPT/Nemotron—it encodes known failure modes so we can compare without an API key; replace it with a live judge before claiming model-level results. Scoped perfect scores on the design suite are partly by construction; the holdout is the honest residual number. ClawForge remains a hackathon-grade system with a simulated runtime for risky ops, not a production multi-tenant broker.

7 Conclusion

Default-deny capability manifests stop unknown tools and force approval on named high-risk actions. They do not stop malicious parameters on tools marked allowed. Parameter scopes close that gap on a design suite and still fail on held-out tricks. If you ship a tool broker, test param abuse explicitly—action-name unit tests will give you a false sense of safety.

Ethics

Attack envelopes are synthetic and intended for defensive evaluation. We do not release working exploit code against production systems. Demo destinations use documentation IPs and example.com-style hosts.

Acknowledgments

ClawForge was built for NVIDIA Hack-a-Claw at UCSC. We thank Baskin Engineering, NVIDIA, and ASUS for the event.

References

Edoardo Debenedetti, Jie Zhang, Mislav Balunović, Luca Beurer-Kellner, Marc Fischer, and Florian Tramèr. 2024. AgentDojo: A dynamic environment to evaluate prompt injection attacks and defenses for LLM agents. In *Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*.

Kai Greshake, Sahar Abdelnabi, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario Fritz. 2023. Not what you've signed up for: Compromising real-world LLM-integrated applications with indirect prompt injection. In *Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security (AISec)*.

Henry M. Levy. 1984. *Capability-Based Computer Systems*. Digital Press.

Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. 2024. Gorilla: Large language model connected with massive APIs. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Fábio Perez and Ian Ribeiro. 2022. Ignore previous prompt: Attack techniques for language models. *arXiv preprint arXiv:2211.09527*.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. 2024. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In *International Conference on Learning Representations (ICLR)*.

Jerome H. Saltzer and Michael D. Schroeder. 1975. The protection of information in computer systems. *Proceedings of the IEEE*, 63(9):1278–1308.

Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein, and Charles E. Youman. 1996. Role-based access control models. *IEEE Computer*, 29(2):38–47.

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

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

Qiusi Zhan, Zhixiang Liang, Zifan Ying, and Daniel Kang. 2024. InjecAgent: Benchmarking indirect prompt injections in tool-integrated large language model agents. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 10471–10506, Bangkok, Thailand. Association for Computational Linguistics.