

USENIX

THE ADVANCED COMPUTING SYSTEMS ASSOCIATION

“Do Not Mention This to the User”: Detecting and Understanding Malicious Agent Skills in the Wild

Yi Liu, Zhihao Chen, and Yanjun Zhang, *Griffith University;* Gelei Deng, *Nanyang Technological University;* Yuekang Li, *University of New South Wales;* Jianting Ning, *Zhejiang Key Laboratory of Digital Fashion and Data Governance, Zhejiang Sci-Tech University;* Leo Yu Zhang, *Griffith University*

https://www.usenix.org/conference/usenixsecurity26/presentation/liu-yi

This artifact appendix is included in the Artifact Appendices to the Proceedings of the 35th USENIX Security Symposium and appends to the paper of the same name that appears in the Proceedings of the 35th USENIX Security Symposium.

August 12–14, 2026 • Baltimore, MD, USA

ISBN 978-1-939133-58-8

Open access to the Artifact Appendices to the Proceedings of the 35th USENIX Security Symposium is sponsored by



جامعة الملك عبدالله للعلوم والتقنية

King Abdullah University of Science and Technology



USENIX Security '26 Artifact Appendix: "Do Not Mention This to the User": Detecting and Understanding Malicious Agent Skills in the Wild

Yi Liu[1,*], Zhihao Chen[1,*], Yanjun Zhang[1], Gelei Deng[2],
Yuekang Li[3], Jianting Ning[4,†], Leo Yu Zhang[1,†]

[1]Griffith University, [2]Nanyang Technological University, [3]University of New South Wales,
[4]Zhejiang Key Laboratory of Digital Fashion and Data Governance, Zhejiang Sci-Tech University

A Artifact Appendix

A.1 Abstract

Reviewers can verify the released dataset (98,380 skills, 157 behaviorally-confirmed malicious) and run a small-batch end-to-end reproduction exercising crawl, static scan, sandboxed dynamic execution, behavioral evidence collection, and optional post-hoc Claude Code–based LLM triage (denoted *CC* hereafter). A separate deterministic analysis suite regenerates the paper's quantitative RQ1/RQ2 results from the released CSVs with no crawl, Docker, or LLM.

A.2 Description & Requirements

The artifact has two top-level components: `data/` holds the released benchmark CSVs, and `code/` holds the reproducibility framework and the deterministic analysis suite. Because the agent-skill ecosystem is live and final malicious labels require manual behavioral verification, the framework defaults to a small-batch experiment; environment variables control larger batches.

A.2.1 Security, privacy, and ethical concerns

1. The released CSVs contain only metadata, with URLs to confirmed-malicious payloads redacted. The framework crawls and runs live agent skills in a monitored Docker container, so run it on a disposable machine or VM.

2. The dynamic executor mounts a host Claude Code credential read-only into the container; because the evaluated skill can influence Claude's behavior, prefer a disposable Claude login.

3. The Docker executor enables tracing and packet capture with additional container privileges; it is suitable for controlled artifact evaluation but is not a strong isolation boundary for arbitrary untrusted code. Captured pcaps and runtime outputs may contain prompts, file paths, or local metadata, so evaluators should not run the artifact on private repositories or with private skills.

4. Per the project README, vulnerabilities found during the study were responsibly disclosed to the affected platform vendors.

A.2.2 How to access

The source is on GitHub at https://github.com/protectskills/MaliciousAgentSkillsBench; the permanent Zenodo archive is https://doi.org/10.5281/zenodo.20285751 (concept DOI, latest release). Reproduce from the evaluated release tag rather than the moving default branch. The Docker sandbox image is `ghcr.io/protectskills/claude-skill-sandbox:lite`, with an offline tarball via the `sandbox-lite-v1` GitHub release for restricted networks.

A.2.3 Hardware dependencies

A modern x86-64 Linux machine with ≥ 4 CPU cores, 8 GB RAM, and 20 GB free disk suffices for the default small-batch experiment; Docker is required for dynamic execution. No GPU is needed—the default Nova-tracer recording mode and the prebuilt `lite` image are CPU-only.

A.2.4 Software dependencies

The host requires Python 3.10+, Docker, Git, `curl`, GNU `timeout` (macOS: expose coreutils `gtimeout` as `timeout`), and a local Claude Code CLI login or an Anthropic-compatible API key (both documented in the README); the optional CC triage additionally needs `jq`. Host-side Python dependencies are in `code/requirements.txt`. The default reproduction pulls the prebuilt sandbox image (≈ 300 MB), so the host needs no `strace`, `tcpdump`, or Nova-tracer hooks—they ship inside the image. A SkillsMP API key is required for the default crawl (free at `skillsmp.com`); a GitHub token is optional but recommended for larger downloads.

*Co-first authors. †Co-corresponding authors.

A.2.5 Benchmarks

The released benchmark, from the paper's January 2026 measurement of two registries (skills.rest, skillsmp.com), consists of:

- `data/skills_dataset.csv`: the Tier-1 snapshot of 98,380 skills across 11,246 repositories, with columns `source`, `repo`, `skill_name`, `classification`, `url`. The `classification` labels are mutually exclusive—`safe` (94,093), `suspicious` (4,130), `malicious` (157)—and `suspicious` + `malicious` = 4,287 reconstructs the paper's nested Tier-2 candidate set. The `url` field is redacted for `malicious` rows and rows sharing their repository.
- `data/malicious_skills.csv`: the curated Tier-3 dataset of 157 behaviorally-confirmed malicious skills (21 from skills.rest, 136 from skillsmp.com) across 69 repositories. Beyond the shared schema it carries a `Pattern` column (semicolon-separated vulnerability-pattern labels) and an aligned per-instance `Severity` column (CRITICAL/HIGH/MEDIUM/LOW).

The default reproduction does not download or execute all 98,380 skills; it runs a configurable small-batch over current SkillsMP results—a few repositories downloaded and statically scanned, with one skill dynamically executed and (if `ENABLE_CC_ANALYSIS=true`) a small post-hoc triage queue.

A.3 Set-up

A.3.1 Installation

From `MaliciousAgentSkillsBench/code/`, create a Python 3.10+ virtual environment, run `pip install -r requirements.txt`, then launch `python3 helper.py`. For first-time setup, step through `1. init` (writes `code/.env`), `2. doctor` (host checks), and `3. image` (pulls the prebuilt image; `load-tar` for the offline tarball, `full-cpu`/`full-custom` for local builds). Menu actions `4. run`, `5. status`, `6. clean` run the small-batch experiment, inspect/edit configuration, and prune outputs.

A.3.2 Basic Test

From `code/`, launch `python3 helper.py` and select `2. doctor`. `SKILLSMP_API_KEY` and the Claude credentials must report `OK` before E2 (the SkillsMP crawl is the only entry into the pipeline). Note that `host strace`/`tcpdump` may remain `MISSING` in some configurations.

A.4 Evaluation workflow

A.4.1 Major Claims

(C1): The released benchmark CSVs load with the documented schema, and the curated Tier-3 dataset contains exactly 157 behaviorally-confirmed malicious skills (21 from skills.rest, 136 from skillsmp.com), matching Table 2 of the paper. This is validated by experiment E1.

(C2): The artifact can run the framework's end-to-end analysis workflow on a bounded sample: crawl, mapping, download, static scan, dynamic execution, and behavioral evidence collection, with optional post-hoc LLM-assisted triage. This is validated by experiment E2.

(C3): Dynamic execution records behavioral evidence needed for manual security analysis, including Claude output, system calls, network traffic, filesystem changes, run metadata, and Nova-tracer session reports. This is validated by experiment E3.

(C4): The released dataset deterministically reproduces the paper's core quantitative results—the attack-technique taxonomy counts, the pattern co-occurrence matrices, and the headline RQ2 hypothesis tests—offline, with no crawl, Docker, or LLM; the paper's RQ3 sophistication and shadow-feature results lie outside the artifact's scope. This is validated by experiment E4.

A.4.2 Experiments

The experiments form two independent tracks. E1 and E4 are deterministic and offline—no credentials, Docker, crawl, or LLM—and reproduce the paper's released-data results; E2 and E3 exercise the live, non-deterministic pipeline on a bounded sample (needing a SkillsMP key, Docker, and a local Claude login or API key). Configuration knobs and batch execution are detailed in the artifact README.

(E1): *Dataset integrity check [10–15 human-minutes].*
How to: From the repository root, run `python3 code/analysis/verify_dataset.py` (stdlib-only), which prints the label counts below; verify the remaining schema and redaction properties by loading the two CSVs (pandas or `csv.DictReader`).
Results: `malicious_skills.csv` contains 157 rows—21 `skills.rest`, 136 `skillsmp.com` (Table 2)—with `Pattern` and `Severity` columns; `skills_dataset.csv` follows the schema `source, repo, skill_name, classification, url` with the mutually exclusive split `safe` 94,093, `suspicious` 4,130, `malicious` 157 (total 98,380), where `suspicious` + `malicious` reconstructs the paper's 4,287 Tier-2 candidates; every malicious row carries a redacted `url`.

(E2): *Default small-batch pipeline [30–60 human-minutes + 30–60 compute-minutes, depending on image-pull, live crawl, and model latency].*
Preparation: Configure `SKILLSMP_API_KEY`, start Docker, log in to Claude Code locally (or configure an Anthropic-compatible API key), and pull (or load) the sandbox image (see Set-up).
Execution: From `code/`, launch `python3 helper.py`

and select `4. run` (the default). The helper lists the six pipeline scripts in order—crawl, mapping, download, static scan, run-queue generation, dynamic execution—and prompts before starting; the optional CC steps are appended only when `ENABLE_CC_ANALYSIS=true`. The crawl-through-scan steps abort on empty input, so on a no-data outcome raise `SKILLSMP_MAX_ITEMS`, `DOWNLOAD_LIMIT`, or `SCAN_LIMIT` via `5. status` → `edit` and re-run. A continuation prompt then runs more pending skills without repeating crawl/download/scan.

Results: Successful execution creates gitignored runtime directories under `code/` (`data/`, `workspace/`, `tasks/`, `logs/`). Dynamic-execution evidence is written under `workspace/dynamic/`, one per-run subdirectory keyed by risk, repo, skill, and run-id (inspected in E3); `scan_results/` appears only when the optional CC steps run. A run that hits `EXEC_TIMEOUT` after a Nova-tracer report is written counts as a captured artifact, not a failure (set `RETRY_TIMEOUT_ARTIFACTS=true` to re-execute it).

(E3): *Dynamic execution evidence inspection [15–30 human-minutes; no additional compute].*

How to: After E2, from `code/`, open the newest run directory under `workspace/dynamic/` (layered by risk, repo, skill, run-id) and inspect its artifacts: the Nova-tracer HTML report in a browser, `network.pcap` in Wireshark or `tshark`.

Results: A successful run contains `claude_output.txt`, `strace.log`, `network.pcap`, `filesystem_changes.json`, `metadata.json`, and a `nova-tracer/` subdirectory with session JSONL traces and an HTML report (`strace.log` requires syscall tracing, which is off by default on macOS hosts, where in-container `ptrace` is unreliable). When `EXEC_REPORT_MODE=true`, the directory also contains a Claude-written `claude_execution_report.md`.

(E4): *Deterministic quantitative reproduction [15–25 human-minutes; no crawl, Docker, or model calls].*

How to: From `code/analysis/`, install its `requirements.txt`, then run `taxonomy_counts.py`, `rq1_landscape.py`, `cooccurrence.py`, and `hypothesis_tests.py` (`code/scripts/ 09–11` wrap the last three). All four read only the released `data/malicious_skills.csv`; `analysis/README.md` maps each script to the paper table or figure it regenerates.

Results: `taxonomy_counts.py` prints the attack-technique taxonomy—632 vulnerability instances across 13 techniques over 157 skills—reproducing Table 4, and `rq1_landscape.py` the Table 5 overview plus the RQ1 vulnerability-density and kill-chain-phase breakdowns. `cooccurrence.py` writes the pattern co-occurrence count, odds-ratio, and conditional-probability matrices and the Figure 6 heatmap, and `hypothesis_tests.py` the Fisher's exact tests with Bonferroni correction and the Mann-Whitney severity test. Outputs are deterministic across runs.

A.5 Notes on Reusability

Larger experiments are configured via `5. status` → `edit`, which updates `code/.env` fields (`SKILLSMP_MAX_ITEMS`, `DOWNLOAD_LIMIT`, `SCAN_LIMIT`, `RUN_QUEUE_LIMIT`, `EXEC_WORKERS`); `6. clean` prunes outputs between iterations. The helper's `exec-dir` batches skills from a local directory without a fresh crawl. Beyond `NOVA_PROFILE=record`, the `scan` and `guard` profiles add Nova warnings or `PreToolUse` blocking. The study's confirmed-malicious skills were removed from the live registries after responsible disclosure, so `code/samples/` ships a defanged copy of one—a calculator hiding a reverse shell, its callback endpoint neutralized to `127.0.0.1`. From `code/`, `samples/run_detection.sh` executes it in the Docker sandbox and `samples/run_cc.sh` runs the CC triage, reproducing an end-to-end detection.

A.6 Threats to Validity

- **Live corpus drift.** The ecosystem changes between runs—repositories disappear, branches move, SkillsMP results vary—so the default crawl will not match the paper's January 2026 snapshot.
- **Skills.rest is unreachable for new crawls.** The registry returns a Cloudflare challenge to headless requests, so the artifact does not re-crawl it; the released CSVs are its only source of skills.rest metadata.
- **Non-deterministic LLM behavior.** Claude Code execution is stochastic: dynamic-execution evidence and optional CC labels are reproducible by category, not byte-for-byte.
- **Manual verification stage.** The 157 "confirmed malicious" labels came from human review of the dynamic evidence. The artifact produces that evidence (E2/E3) but does not auto-label, so the count cannot be mechanically re-derived from a small-batch run.

A.7 Version

Based on the LaTeX template for Artifact Evaluation V20231005. Submission, reviewing and badging methodology followed for the evaluation of this artifact can be found at https://secartifacts.github.io/usenixsec2026/.