SRE-Zero: Environment-Grounded Evaluation for Reliable Tool-Using Agents

Devaansh Pathak

Draft v1 – 19/05/2026

Abstract

Large language model agents are increasingly evaluated on their ability to use tools, inspect external state, and complete multi-step workflows. However, many benchmarks still emphasize final answer correctness rather than the reliability of the intermediate process. This draft introduces SRE-Zero, an early-stage simulation benchmark for studying reliable tool-using agents in site reliability engineering incident-response workflows. Agents must diagnose simulated infrastructure incidents, inspect service logs and metrics, apply minimal remediations, and submit a final resolution under a step budget. The environment uses structured actions, typed observations, deterministic task configurations, partial-credit rewards, and trajectory-level metrics. Preliminary results from a small first sweep suggest that simple prompting and ReAct-style baselines can gather relevant evidence but often fail to convert evidence into correct remediation and resolution. These early results are not intended as model rankings; instead, they motivate SRE-Zero as a controlled testbed for measuring diagnosis, action validity, remediation quality, and failure modes separately.

1 Introduction

Tool-using agents are often presented as systems that can reason, inspect external context, and act in software environments. In practice, reliability is not only a question of whether the final answer is correct. A useful operational agent must choose appropriate tools, gather relevant evidence, avoid irrelevant or unsafe actions, apply minimal fixes, and know when an incident has actually been resolved.

Incident response is a useful setting for evaluating these properties. A site reliability engineer does not usually resolve an outage by guessing a final answer. The process is sequential: inspect symptoms, identify evidence, form a root-cause hypothesis, remediate the smallest plausible failure, and verify that the incident is resolved. This makes incident response a natural environment for studying reliable tool use.

SRE-Zero is an early-stage benchmark for simulated incident response. It is simulation-only and does not control real infrastructure. The current repository implements a deterministic Python environment, a Gym-style wrapper, baseline agents, task configurations, evaluation scripts, metrics, and a small frontend console. The benchmark is designed to grow toward a paper-quality suite while remaining reproducible and inspectable.

Contributions. This draft makes four initial contributions:

1. A typed, deterministic incident-response environment with structured actions and observations.

2. A task suite spanning web, database, cache, message queue, and load balancer incidents.
3. A partial-credit reward function and metrics that separate evidence gathering, remediation, success, efficiency, invalid actions, wrong fixes, and distractor failures.
4. Preliminary baseline results showing separable failure modes in simple tool-using LLM agents.

2 Related Work

This section is intentionally incomplete in the first draft. The final paper should position SRE-Zero relative to:

- tool-use and agent benchmarks,
- reinforcement-learning environments for language agents,
- software engineering and debugging benchmarks,
- operations and incident-response evaluation,
- reliability, calibration, and process-based evaluation of LLM agents.

The central distinction to develop is that SRE-Zero evaluates agents inside a stateful environment where intermediate tool use, evidence coverage, and remediation behavior are first-class evaluation targets.

3 Environment

SRE-Zero models a small production-like service graph. The current environment contains five simulated services:

- `web_server`
- `database`
- `cache`
- `message_queue`
- `load_balancer`

Each service has status, logs, metrics, configuration, and dependency metadata. All state is in memory. Actions never execute shell commands and never affect real systems.

3.1 Observation Space

At each step, the agent receives an observation containing the incident id, step count, remaining step budget, alert text, last action, last action result, known findings, available tools, and a done flag. Hidden fields such as root cause, correct fix, and reward breakdown are not included in the agent-visible observation.

3.2 Action Space

Agents use structured function-style actions:

```
inspect_logs(service)
inspect_metrics(service)
check_status(service)
inspect_config(service, key?)
restart_service(service)
update_config(service, key, value)
resolve_incident(root_cause, fix)
escalate(reason)
```

Actions can be submitted as Pydantic objects or strings. Invalid actions return controlled error observations and penalties instead of crashing the environment.

3.3 Episode Termination

An episode ends when the incident is correctly resolved, the step budget is exhausted, the agent escalates, or an incorrect final resolution terminates the task. Each task has a deterministic step budget, usually eight actions.

4 Task Suite

The expanded suite contains 25 deterministic incident tasks. Task definitions are JSON configs with hidden root causes, visible alerts, relevant evidence, service patches, expected scripted trajectories, distractors, and correct remediation validators.

Table 1: Current SRE-Zero task splits.

Split	Count	Description
Easy	7	Direct failures and simple configuration issues
Medium	10	Multi-signal diagnosis and moderate configuration fixes
Hard	8	Misleading symptoms, distractors, and cross-service root causes

The first result sweep discussed later used an earlier 15-task version of the suite. The environment has since been expanded to 25 tasks. The results should therefore be read as preliminary evidence about the evaluation design rather than final performance on the expanded benchmark.

5 Rewards and Metrics

The reward function is normalized to $[0, 1]$ at the episode level and returns per-step shaping rewards. Positive reward components are:

Penalties are applied for invalid actions, repeated invalid actions, wrong remediations, premature resolution, unrelated restarts, and repeated useless actions.

SRE-Zero reports the following aggregate metrics:

- success rate,

Table 2: Reward components.

Component	Maximum
Relevant evidence gathered	0.20
Correct root cause identified	0.25
Correct remediation applied	0.25
Correct resolution submitted	0.20
Efficiency bonus	0.10

- mean reward,
- mean steps to termination,
- invalid action rate,
- evidence coverage,
- wrong remediation rate,
- premature resolution rate,
- distractor failure rate.

The distractor failure metric counts wrong remediations against task-configured distractor services. This separates generic wrong fixes from failures caused by plausible but misleading symptoms.

6 Baselines

The repository currently includes six baseline categories:

- **Random**: samples random structured actions and sometimes emits invalid services.
- **Scripted expert**: follows each task's expected action pattern. This is an upper-bound baseline and is allowed to use task-specific policy knowledge.
- **Prompting**: calls an OpenAI-compatible chat endpoint with the current observation and asks for one action.
- **ReAct**: keeps a compact thought/action history across the episode.
- **Open-source LLM profile**: prompting-style profile for hosted or local OpenAI-compatible open-source models.
- **Frontier profile**: ReAct-style profile for stronger hosted models.

7 Preliminary Experiment

The first cheap sweep was run before the 25-task expansion, on the 15-task suite. It used five episodes per task for deterministic baselines and one episode per task for LLM baselines. The run used seed 0 and a 90-second timeout.

Table 3: Preliminary cheap sweep results on the earlier 15-task suite.

Baseline	Model	Marks	Success	Reward	Evidence	Invalid	Steps	Errors
scripted	deterministic/scripted	93.7	1.00	0.946	1.00	0.00	4.47	0
frontier	openai/gpt-5.5	67.7	0.73	0.603	0.83	0.01	6.40	1
react	openai/gpt-5-mini	21.5	0.00	0.040	0.81	0.15	4.40	13
prompting	openai/gpt-5-mini	17.7	0.00	0.000	0.63	0.00	3.73	12
open-source	ibm-granite/granite-4.1-8b	16.6	0.00	0.010	0.57	0.00	8.20	0
random	deterministic/random	5.6	0.00	0.001	0.08	0.21	3.69	0

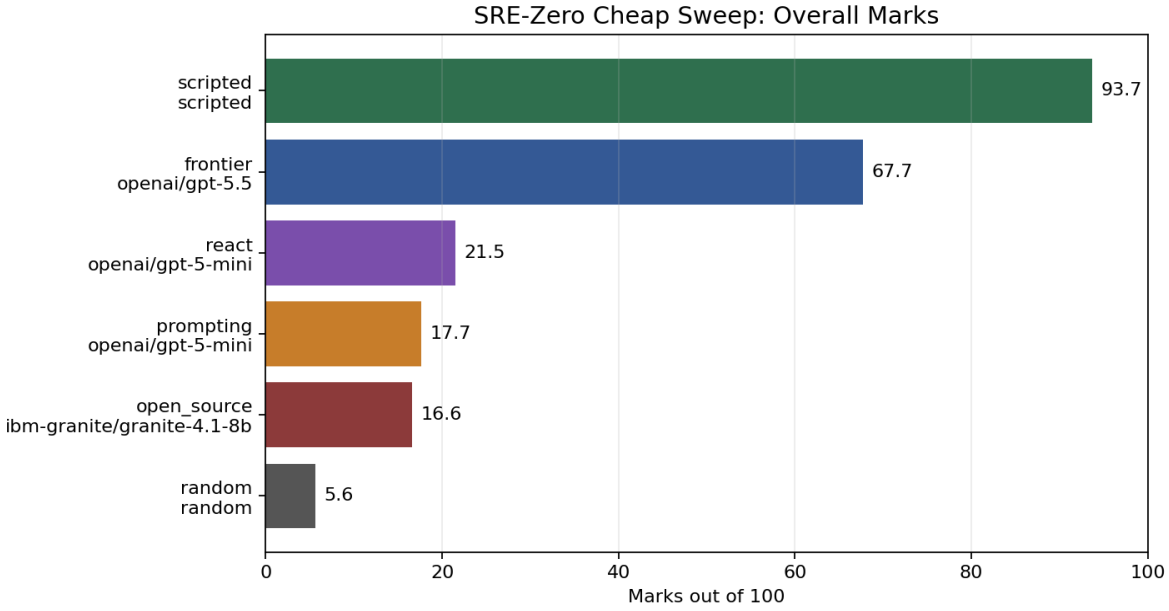

Figure 1: Overall marks from the first cheap sweep.

8 Failure Analysis

The most useful early signal is the gap between evidence coverage and success. The ReAct-style `openai/gpt-5-mini` run reached 0.81 evidence coverage but had zero task success. The prompting run reached 0.63 evidence coverage and also had zero success. This suggests that these agents are not merely failing to call tools. They often inspect plausible logs and metrics, but fail to convert evidence into a correct remediation and final resolution.

The frontier profile solved 11 of 15 tasks in the preliminary sweep. Its failures were concentrated in tasks requiring precise database remediation or careful final resolution. Several failed tasks still had high evidence coverage, which again supports the view that diagnosis and remediation should be measured separately.

Provider reliability also appears in the trajectories. Several LLM runs recorded errors where the chat-completions response contained null message content. In SRE-Zero, these are counted as agent or provider failures rather than crashing the entire evaluation.

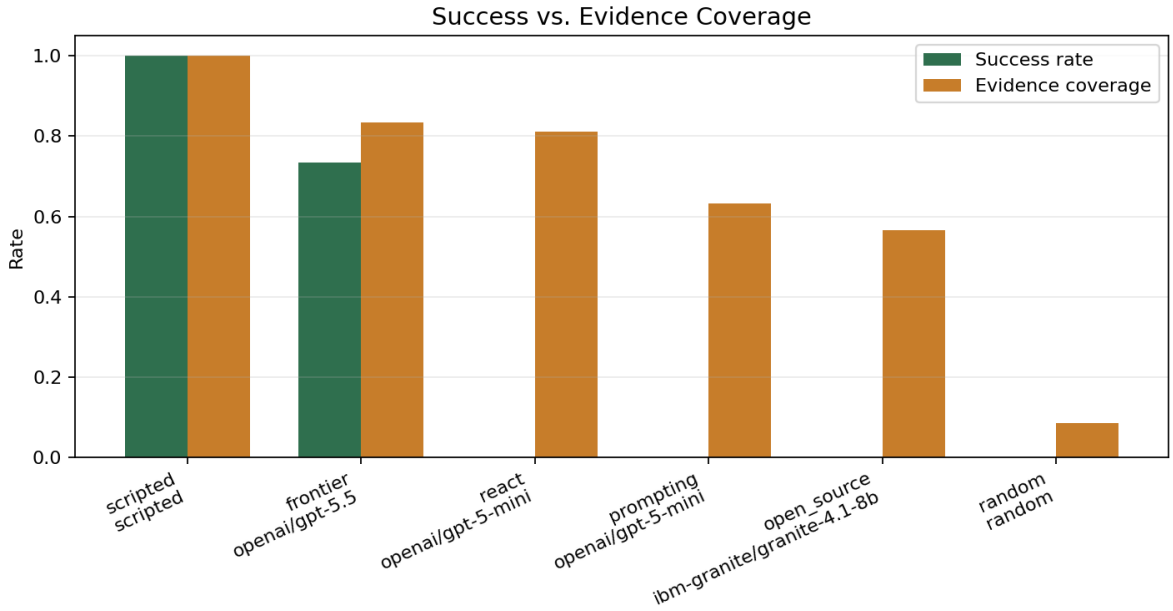

Figure 2: Evidence coverage can be high even when final success is zero.

9 Limitations

This draft has several important limitations:

- The preliminary results use the earlier 15-task suite, not the expanded 25-task suite.
- LLM baselines were run for only one episode per task.
- Only one seed was used.
- The model set is small and provider-dependent.
- The scripted expert is an upper bound, not a fair general agent.
- The simulator is deterministic and does not yet model richer side effects or recovery verification.

The correct interpretation is therefore not that one model is broadly better than another. The appropriate interpretation is that SRE-Zero can expose intermediate reliability failures that are hidden by final-answer-only evaluation.

10 Next Steps

The next version should evaluate the expanded 25-task suite, repeat runs across seeds, run multiple episodes per LLM baseline, add cost and token accounting, separate provider errors from reasoning failures, and strengthen the task suite with additional cross-service incidents.

11 Conclusion

SRE-Zero is an early benchmark for studying reliable tool use in simulated incident response. The environment is intentionally small, safe, deterministic, and inspectable. Preliminary results suggest that simple LLM baselines can gather evidence without reliably resolving incidents. This makes the benchmark useful for studying not only whether agents succeed, but how and where they fail.