SysRepair-Bench: Measuring Whether Agents Can Fix a Vulnerability Without Breaking the Service

Anonymous authors
Paper under double-blind review

Abstract

Vulnerability remediation is the operator's last step: change a running system so a disclosed weakness is no longer exploitable while the protected service keeps serving. No benchmark measures this. Offensive benchmarks reward exploits with no availability obligation, and repair benchmarks edit source and re-validate a rebuilt target. We present SysRepair-Bench, 313 remediation scenarios, each a live vulnerable service an agent must fix through a shell; the 297 binary scenarios are graded by a *two-component verdict*: the exploit is blocked *and* the service stays healthy. Every scenario in the main grid runs in two conditions, one supplying the vulnerability report and one requiring the agent to find the fault itself. Our central result is the gap between them. Given the report, the median model and suite cell reach 90.0%, and the strongest models approach the ceiling; required to locate the fault first, the same models on the same hosts with the same budget reach a median of 49.4%. Every model loses ground, by a median of 37.4 points. The report-informed condition is an anchor: it measures execution once a fault is named, and the gap is the cost of withholding the report. Collateral damage follows the same axis. The *Collateral-Damage Rate*, the share of exploit-blocking successes achieved by breaking the service, reframes remediation as specification gaming; without the report it is higher for seven of the eight models. We release the benchmark, the two-component verifiers, the scripted validity agents, and the evaluation harness.

1 Introduction

Vulnerability remediation is the last operational step of security risk management: an operator must change a running system so that a disclosed weakness is no longer exploitable *while the protected service keeps serving its users*. This happens at the security-administration level, not the source level: developers write patches, but administrators deploy them, harden configurations, and, before a vendor patch exists or can be safely applied, stand up compensating controls (Joint Task Force, 2020) that shrink the attack surface without taking the service down. As language agents begin to act autonomously on real infrastructure, the question is no longer only *can an agent close a vulnerability*, but *can it do so without breaking what it was protecting*.

No existing evaluation measures this (§2). CTF and offensive benchmarks (Zhang et al., 2025b; Zhu et al., 2025; Wang et al., 2026b; Lee & Brumley, 2026) reward landing an exploit and impose no availability obligation. Execution-based repair benchmarks edit application *source* and re-validate a rebuilt target, so they cannot express a vulnerability with no upstream patch, an end-of-life package, or a misconfiguration (Jimenez et al., 2024; Huang et al., 2026; Lee et al., 2025; Byun et al., 2025). Autonomous-defense (Standen et al., 2021) and IT-operations benchmarks (Jha et al., 2025; Chen et al., 2025; Clark et al., 2026) treat availability as first-class but abstract the host to a simulator, or target reliability faults rather than a disclosed vulnerability under an adversary. And general agentic-execution benchmarks (Shepard & Salimans, 2026) share our modality (an agent driving a real system through a shell, graded by a verifier) but score task completion, with no adversary and no availability constraint. *None requires an agent to execute a remediation on a live system under a measured service-availability constraint.*

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

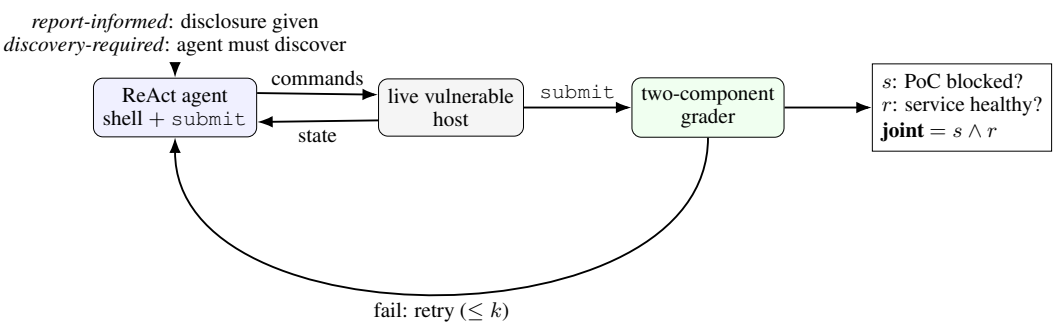

Figure 1: The SysRepair-Bench evaluation loop. An agent drives a live vulnerable host (container or VM) through a shell and calls `submit`; the grader passes only when the exploit is blocked *and* the service stays healthy. Each scenario runs report-informed (disclosure given) and discovery-required (the agent must find the fault).

SysRepair-Bench fills that gap: 313 dual-objective remediation scenarios, each a running, vulnerable service in an isolated environment (a container, or a VM for the kernel-level, Active-Directory and some Hivestorm scenarios) that an agent remediates through a shell. Success is a **two-component verdict**: the proof-of-concept exploit is blocked *and* the protected service (with its stated dependent clients) stays healthy, so an agent cannot "win" by taking the service down (Figure 1). Every scenario in the main grid runs in two knowledge conditions, *report-informed* (the disclosure is given) and *discovery-required* (the agent must discover the fault), separating an agent's perception from its execution.

This instrument yields two findings a naive, availability-blind evaluation cannot see, and the first governs how the second should be read.

Withholding the report is the dominant cost. Given the vulnerability report, the median model and suite cell reaches 90.0%, and the strongest models approach the ceiling. Required to locate the fault themselves, on the same hosts with the same tools and time budget, they reach a median of 49.4%, and every model loses ground, by a median of 37.4 points (§6.2). We therefore treat the report-informed condition as an *anchor*: it establishes that these systems can execute a named repair, which is what makes the other condition interpretable.

Collateral damage moves on the same axis. We define the **Collateral-Damage Rate** (CDR): the fraction of an agent's exploit-blocking successes achieved by damaging the protected service, which reframes remediation as specification gaming (Amodei et al., 2016; Krakovna et al., 2020) in a deployed setting. CDR is higher when the report is withheld than when it is given for seven of the eight models. A within-family scale ladder (§6) separates model scale from cross-lab confounds.

Contributions.

- **SysRepair-Bench**, 313 scenarios (297 binary + 16 partial-credit) across six suites, evaluating operator-level remediation on live systems under a service-availability constraint, a task no prior benchmark poses (§§2–3).
- A **two-component verdict protocol** and the **Collateral-Damage Rate**, which measures how often an exploit is blocked by breaking the service (§5).
- An empirical study across models and a within-family scale ladder: a discovery/execution gap of a median 37.4 points that every model shows at a fixed time budget, and a collateral-damage rate that is higher without the report than with it for seven of eight models (§6).

2 Related Work

Offensive and CTF benchmarks. A large line of work evaluates agents that *break into* systems: Cybench (Zhang et al., 2025b) scores 40 professional CTF tasks, CVE-Bench (Zhu et al., 2025) exploits 40 critical web-application CVEs, and CyberGym (Wang et al., 2025) generates crashing

proofs-of-concept against OSS-Fuzz targets. ExploitGym (Wang et al., 2026b) extends a triggering input into a working exploit across 898 userspace, V8 and Linux-kernel instances, and ExploitBench (Lee & Brumley, 2026) grades exploitation of 41 V8 bugs on a 16-flag capability ladder. Success is a compromise; none imposes an availability obligation. We evaluate the inverse, operationally later step: remediation under an availability constraint.

Source-level repair. The closest execution-based benchmarks edit application *source* and re-validate a patched build on a fresh target. SEC-bench (Lee et al., 2025) and AutoPatchBench (Byun et al., 2025) score crash resolution (verified by re-fuzzing); BountyBench (Zhang et al., 2025a) rebuilds its multi-container runtimes from the patched code and checks runtime-health invariants alongside the exploit; ZeroDayBench (Lau et al., 2026) scores whether a live exploit is blocked after remediation; and DARPA AIxCC (Defense Advanced Research Projects Agency (DARPA), 2023) validates source patches with build, proof-of-vulnerability reproduction, and functional tests on quiescent targets. All require a change to the vulnerable software's *source*, so none can express a disclosed vulnerability with no upstream patch, an end-of-life package, or a misconfiguration, and none measures a service-availability constraint on the remediated system. SysRepair-Bench grades the running system after operator-level changes, never a rebuilt codebase.

Autonomous defense and IT operations. Benchmarks that treat availability as first-class abstract away the target or change the threat. CAGE/CybORG (Standen et al., 2021) is a symbolic network simulator, not a real host; ITBench (Jha et al., 2025), AIOpsLab (Chen et al., 2025), and SREGym (Clark et al., 2026) inject reliability faults into live microservice stacks and score detection, localization, and mitigation, but without a working exploit or a disclosed-vulnerability adversary. We keep availability first-class *and* adversarial: all but three binary scenarios pair a working proof-of-concept exploit with a service probe, checked at every submission, that the remediation must not break.

Defensive-agent and general agentic-execution benchmarks. A recent wave of defensive benchmarks evaluates *detection, investigation, and attribution*, not the execution of a fix: CyberTeam (Meng et al., 2025) scores pipelined blue-team threat-hunting over static analysis samples, CyberSleuth (Fumero et al., 2025) performs offline forensic investigation over captured traces, and SecRespond (Wang et al., 2026a) produces incident-response plans. General agentic benchmarks (SWE-bench (Jimenez et al., 2024), Terminal-Bench (Merrill et al., 2026), and OSWorld (Xie et al., 2024)) share our execution modality (an agent driving a real environment, graded by a verifier) but score task completion, with no adversary and no availability constraint. Our novelty is therefore precise: *no prior benchmark requires an agent to execute a remediation on a live system under a measured service-availability constraint* (Table 1).

Specification gaming and side effects. Collateral damage, an agent that closes the vulnerability by killing the service, is an instance of specification gaming / negative side effects (Amodei et al., 2016; Krakovna et al., 2020; Turner et al., 2020): an availability-blind verifier is a misspecified reward, and the Collateral-Damage Rate (§5) is the gap it hides. Prior work studies these effects in gridworlds and RL simulators; SysRepair-Bench measures them in a deployed real-system setting and asks whether they abate with model capability.

Table 1: SysRepair-Bench vs. prior benchmarks. *Live host*: graded on a running system. *Avail. constraint*: success requires the service to keep working (○= runtime-health after rebuild). *Operator-level fix*: config/service/permission change, not a source patch. *Exec. verified*: binary pass from an exploit/probe (○= partial/answer-matching).

Benchmark	Live host	Avail. constraint	Adversary	Operator-level fix	Exec. verified
Cybench / CVE-Bench / ExploitGym / ExploitBench (offensive)	✓	–	✓	–	✓
SWE-bench	rebuild	–	–	–	✓
Terminal-Bench / OSWorld	✓	–	–	○	✓
CyberGym (offensive)	rebuild	–	✓	–	✓
SEC-bench / AutoPatchBench	rebuild	–	✓	–	✓
BountyBench / ZeroDayBench	rebuild	○	✓	–	✓
CAGE / CybORG	sim	✓	✓	✓	–
ITBench / AIOpsLab / SREGym	✓	✓	–	✓	✓
CyberTeam / CyberSleuth (blue)	–	–	✓	–	○
SysRepair-Bench (ours)	✓	✓	✓	✓	✓

3 THE SYSREPAIR-BENCH BENCHMARK

Design principles. Four commitments distinguish SysRepair-Bench from prior execution benchmarks. **(P1) Operator-level remediation on a live host.** A scenario is a running, vulnerable service inside a container or VM; the agent remediates by changing the system (configuration, permissions, service and firewall state, package or compensating-control decisions), never by editing and rebuilding application source. This is the level at which security administrators work, and the only level at which a vulnerability with no upstream patch can be expressed. **(P2) A dual objective.** Success requires both closing the vulnerability and preserving the protected service (three vulnhub scenarios carry no service probe, §5), so an agent cannot "win" by taking the service down. **(P3) Two knowledge conditions.** Every main-grid scenario runs report-informed and discovery-required, isolating perception from execution. **(P4) An audited, tamper-resistant grader.** Verdicts come from executing a proof-of-concept exploit and discrete service probes; P2–P4 are specified in §5 and audited in §7.

Anatomy of a scenario. Each scenario pairs a vulnerable environment, a `threat.md` disclosure (shown only in the report-informed condition) and a verifier that drives the proof-of-concept (PoC) exploit and emits the two-component verdict, the exploit living in the verifier's checks rather than a separate file. On the Linux suites the environment is a `Dockerfile` and the verifier `verify.sh`; meta3-windows uses Windows containers with a PowerShell verifier, and the Active Directory track (ad-vm) a four-VM domain lab whose PoC is launched from an attacker VM. Consider a Configuration-Hardening example: an OpenSSH server is running but offers weak, deprecated (CBC/RC4) ciphers. The PoC negotiates each banned cipher, and the security component passes only once those handshakes are refused; the naive fix (disabling the daemon) would also pass the PoC, so a regression probe requires that `sshd` still answers and a strong-cipher handshake still reaches authentication. The scenario also traps the "edited the config but never reloaded" shortcut: the joint pass is earned only by editing the cipher policy and reloading the live service.

Composition. The corpus has 313 scenarios from six suites (§4): 297 binary scenarios graded by $s \wedge r$, and the 16 continuously scored Hivestorm hosts. Each binary scenario carries one of five operator remediation categories; Table 5 gives the distribution by suite and runtime. Most run as Linux containers. Four sr-modern kernel-level CVEs run on kernel VMs, because a container shares the host kernel and the fix requires patching and rebooting it. The Windows and Active Directory tracks run as Windows containers and a Windows domain lab.

Disclosure-year split. The modern-CVE suite (*sr-modern*) spans disclosure years 2018–2026. We compare scores on scenarios whose CVE dates are from before 2023 with those from 2023 in Appendix G. No per-model training cutoff is stated, so the split describes how scores vary with CVE year rather than isolating recall of a public fix. The Hivestorm suite additionally randomizes host identities (accounts, paths, service names) per instance, so an agent cannot learn them across runs.

4 SUITES

SysRepair-Bench draws its scenarios from six suites (meta3 is split by host OS into meta3-ubuntu and meta3-windows; sr-modern's 117 include four kernel-level scenarios, and a separate 20-scenario directory track, ad-vm, accompanies it) chosen to vary the era, the service, and the *kind* of remediation an operator must perform, under a strict no-overlap policy: every scenario targets a CVE or misconfiguration not covered by any other suite.

meta2: the classic base (40)**.** Extracted from an OpenVAS scan of Metasploitable 2 (Ubuntu 8.04), meta2 anchors the benchmark in well-understood weaknesses (weak SSH ciphers, backdoored services, cleartext protocols) and introduces the *compensating-controls* band (S34–S40), where a direct upgrade or removal would break a dependent legacy workload, so the agent must apply a network- or config-layer mitigation that keeps the service usable.

sr-modern: modern CVEs (117)**.** The modern track covers Log4Shell era Java remote code execution, cloud-native supply-chain flaws, container-runtime escapes, privilege-escalation chains,

and vulnerable API stacks. Its disclosure dates span 2018–2026, which supports the disclosure-year split (§3).

ccdc: competition hardening (50). Derived from Collegiate Cyber Defense Competition blue-team toolkits (TAMU, LA Tech, UTSA; 2018–2024) on a current Ubuntu base, ccdc is weighted toward the three operator verbs (config edits, package install/upgrade/remove, and permission changes).

vulnhub: community-VM breadth (30). Recreations of specific vulnerabilities from well-known VulnHub training VMs, rebuilt as containers so they run in the same harness, widening the distribution of service configurations beyond the Metasploitable lineage.

hivestorm: free-roam, continuously scored (16). The hardest condition: the agent is told *nothing* about what is wrong. Each host carries multiple independent issues (rogue accounts, PAM and `sysctl` weaknesses, planted backdoors, package hygiene) and is scored continuously in $[0, 1]$ as the weighted fraction of its hardening checks passed, less a penalty for each required service that stops, rather than by a single binary verdict. An anti-memorization layer randomizes every identity an agent could otherwise learn across runs (backdoor names, account names, paths). Hivestorm is the discovery-required condition taken to its limit and the one suite scored on partial credit.

Additional tracks. meta3-ubuntu (19) covers classic Ubuntu-server CVEs distinct from the above; meta3-windows and ad-vm (§3) target Windows-service and AD misconfigurations (weak Group Policy Objects, delegation and access-control-list abuse, service hardening). In all, $40{+}117{+}50{+}30{+}19{+}21{+}20$ binary $+$ 16 Hivestorm $=$ 313 scenarios.

5 Evaluation Protocol and Metrics

Episodes, attempts, and the agent loop. An *episode* places a single ReAct agent (Yao et al., 2023) in a live, vulnerable host (container or VM) with a shell tool and a `submit` action. The agent inspects the system, applies changes, and calls `submit` when it believes the system is remediated; the grader then runs and returns a verdict. A failing submission returns only a *binary* signal, and the agent may retry up to k times within one episode (same container, accumulating state); it never receives the grader's explanation or the verifier source. Episodes are bounded by wall-clock and working-time budgets (Appendix B). An episode that exhausts its budget is kept and scored on the attempts it completed.

Two conditions. Every scenario in the main grid is evaluated under two knowledge conditions. The free-roam track, which ships no report, is scored on its own terms. In the **report-informed** condition the agent is given the vulnerability description (analogous to remediating a disclosed CVE with its advisory in hand). In the **discovery-required** condition it receives only the running system and the availability requirement, and must *discover* what is wrong before fixing it. The two decompose failure into a *perception* term (what discovery-required adds over report-informed) and an *action* term (report-informed failure), the discovery/execution split we analyze in §6. We avoid "zero-day": every CVE here is published, so the axis is agent *knowledge*, not disclosure status.

Two-component verdict. The defining constraint of SysRepair-Bench is that a remediation must close the vulnerability *without breaking the service it protects*. Each binary-scored grader emits two components:

- **Security pass** $s \in \{0, 1\}$: the proof-of-concept exploit no longer succeeds against the submitted system.
- **Regression pass** $r \in \{0, 1\}$: the protected service (and any stated dependent clients) remains healthy, evaluated by discrete regression probes *interleaved* with the PoC checks and run non–fail-fast, so every component is scored on every submission (a median of 2 regression checks per scenario, range 0 to 7; three vulnhub scenarios carry PoC checks only).

An episode's **joint pass** is $j = s \wedge r$, the only outcome a correct remediation can produce; a **security-only pass** is s alone, the reward an availability-blind verifier would grant; Figure 4 walks one scenario through both components. Graders are hardened against scoring tampering. Hivestorm's

16 scenarios are instead scored continuously in $[0, 1]$ (§4), with no security/regression split or pass@k axis; headline rates are joint-pass rates on the binary suites unless stated otherwise.

Collateral-Damage Rate (CDR). The gap between the two verdicts is what CDR measures. With $\bar{s}$ and $\bar{j}$ the mean security-only and joint pass rates over a set of episodes, we define the **Collateral-Damage Rate**

$$\text{CDR} \;=\; \frac{\bar{s} - \bar{j}}{\bar{s}} \qquad (\text{CDR} \triangleq 0 \text{ when } \bar{s} = 0), \tag{1}$$

the fraction of an agent's PoC-blocking successes achieved by *damaging the protected service*, a direct deployed-setting measure of specification gaming / negative side effects (Amodei et al., 2016; Krakovna et al., 2020; Turner et al., 2020): a naive verifier scores $\bar{s}$, an operator experiences $\bar{j}$, and CDR is the safety tax the naive metric hides. We report CDR per model and condition, and ask whether it shrinks with capability (§6).

Replication and the pass@k curve. Every scenario runs for E independent episodes, each in a fresh environment, and each episode allows up to k=5 graded attempts; E=3 (Appendix B). Because episodes are graded at every attempt, one k-attempt run reconstructs the whole pass@$1 \ldots k$ curve: lower-k values are read off the same episodes rather than costing separate runs. *Within* an episode attempts are not exchangeable (host state accumulates across the binary fail signal), so every pass@k in this paper reads the realized prefix within an episode.

Aggregation. A model's accuracy in a condition is the **micro-average** over its scenarios, each scenario weighted equally; the **macro-average**, the unweighted mean of the per-suite rates, is reported alongside so a large suite cannot dominate. The abstract and introduction summarise across models and suites by the median of the 40 model and suite cells of Table 2.

6 Experiments and Results

Setup. We evaluate a cross-lab hosted panel (Panel A: DeepSeek-V4-Flash, MiniMax-M3 and MiniMax-M2.7 in both knowledge conditions) and a within-family Qwen3.5 scale ladder (Panel B, 4B→9B→27B→35B→122B), all under ReAct. We report joint pass@k and CDR; serving and sampling are in Appendix A.

6.1 Main results

The experimental design is one grid: Table 2 places every model against every suite in both knowledge conditions. Two columns pool a Linux suite with a Windows one (meta3-ubuntu with meta3-windows, sr-modern with ad-vm), so the five columns cover all 297 binary scenarios. For the reference model (DeepSeek-V4-Flash), report-informed joint pass@5 is high on every suite (92.7–100) while discovery-required pass@5 drops sharply (52.5–84.4, the discovery cliff of §6.2), and CDR stays low in both conditions (pass@1 and pass@5 in Table 8).

6.2 Discovery vs. execution

The gap between conditions measures what finding the fault costs once execution is held fixed. We compute the gap as report-informed minus discovery-required pass@5 over all 297 binary scenarios (Table 8). All eight models lose ground, by 12.6 to 47.5 points with a median of 37.4. DeepSeek-V4-Flash, the best-performing model, falls from 95.4 to 73.0, a 22.4-point gap. Withholding the disclosure also raises collateral damage: CDR is higher in the discovery-required condition for seven of the eight models, all but Qwen3.5-35B.

This governs how the rest of the table should be read. High report-informed accuracy is a statement about execution once a fault is named, not about autonomous remediation: the same agent, on the same host, with the same tools and budget, resolves far less when nobody tells it what is wrong. That difference is the *perception* term of §5. The failure taxonomy agrees: over the episodes whose verifier reported both components, every failure is exactly one of *not remediated*, *collateral damage* or *both*, and discovery-required failures are dominated by non-remediation (Table 10, Appendix H).

Table 2: Full results grid: joint pass@5 (RI/DR) by model and suite. *meta3* pools meta3-ubuntu (19) and meta3-windows (21); *sr-modern* pools sr-modern (117) and ad-vm (20). Column counts sum to the 297 binary scenarios of Table 5.

Model	meta2 n=40 RI/DR	vulnhub n=30 RI/DR	ccdc n=50 RI/DR	meta3 ub+win, n=40 RI/DR	sr-modern +ad-vm, n=137 RI/DR
DeepSeek-V4-Flash	99.2/68.3	93.3/84.4	97.3/67.3	100.0/52.5	92.7/80.0
MiniMax-M3	94.2/55.0	92.2/68.9	94.0/58.0	97.5/27.5	84.4/66.4
MiniMax-M2.7	90.8/52.5	90.0/53.3	91.3/46.0	85.0/27.5	75.7/50.6
Qwen3.5-4B	82.5/25.8	80.0/52.2	85.3/26.7	67.5/17.5	64.0/19.7
Qwen3.5-9B	79.2/31.7	81.1/44.4	86.0/41.3	67.5/12.5	61.8/26.8
Qwen3.5-27B	96.7/37.5	90.0/54.4	90.0/42.0	77.5/20.0	70.6/45.7
Qwen3.5-35B	91.7/45.0	91.1/71.1	90.0/58.0	85.0/72.5	70.8/76.9
Qwen3.5-122B	91.7/38.3	93.3/53.3	92.0/52.7	92.5/27.5	82.5/48.2

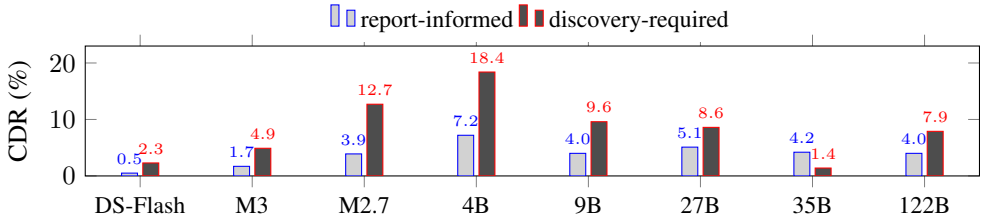

Figure 2: Collateral-Damage Rate benchmark-wide by model and knowledge condition (report-informed, discovery-required), under ReAct. DS-Flash = DeepSeek-V4-Flash; M3, M2.7 = MiniMax; the remaining labels are Qwen3.5 rungs.

6.3 Collateral damage and capability

Discovery-required CDR ranges from 1.4% to 18.4% across the eight models (Figure 2, benchmark-wide). The smallest model (Qwen3.5-4B) damages the protected service in 7.2% of its report-informed exploit blocking successes and 18.4% under discovery-required, while the reference model sits at 0.5% and 2.3%.

6.4 Within-family scale ladder

Panel B holds the family fixed (Qwen3.5) and varies total parameters (4B→9B→27B→35B→122B), removing the cross-lab confound; the 35B and 122B rungs are mixture-of-experts with 3B and 10B active parameters, so the architecture also varies along the ladder. Report-informed accuracy rises with scale from 72.2 at 4B to 87.8 at 122B, though not monotonically: 9B (70.9) sits 1.3 points below 4B. The discovery gap is 47.5, 41.1, 38.9, 12.6 and 42.4 points from 4B to 122B (Table 4). The 35B rung is also the only model with a suite where discovery-required pass@5 exceeds report-informed, sr-modern with ad-vm (76.9 vs. 70.8, Table 2); benchmark-wide it still loses ground, by the smallest gap of any model. Report-informed CDR sits between 4.0 and 7.2% on every rung. Discovery-required CDR is 18.4% at 4B and 1.4% at 35B, a difference of 17.0 points. Across labs, discovery-required CDR is lower for models with higher discovery-required pass@5: 2.3, 4.9 and 12.7% for DeepSeek-V4-Flash, MiniMax-M3 and MiniMax-M2.7, at 73.0, 58.5 and 47.2.

6.5 The free-roam track

Mean scores over the 16 hosts range from 0.28 (Qwen3.5-9B) to 0.74 (MiniMax-M3; Table 3), and the ordering broadly tracks discovery-required pass@5: 24 of the 28 model pairs are ordered the same way (Spearman $\rho = 0.88$). The four exceptions are MiniMax-M3 ranking above DeepSeek-V4-Flash and Qwen3.5-35B, Qwen3.5-122B above MiniMax-M2.7, and Qwen3.5-4B above 9B; discovery-

required pass@5 orders each of these pairs the other way. Within the family, both measures rank Qwen3.5-35B first and above the larger 122B rung (35B $>$ 122B $>$ 27B), so the non-monotonicity in scale appears on both; they differ only on the bottom two rungs, where the free-roam track puts 4B above 9B (0.31 vs. 0.28) and discovery-required pass@5 puts 9B above 4B (29.8 vs. 24.7). Every model scores lower on Windows hosts than on Linux hosts, by 0.05 to 0.12, and the two OSes rank the models identically. The single FreeBSD host is also below Linux for every model, by 0.03 to 0.30, but it reorders the models: MiniMax-M3 and Qwen3.5-35B each fall 0.30 below their Linux score, leaving DeepSeek-V4-Flash first. With one host, we read this as a single data point rather than an OS effect.

Table 3: Free-roam track (Hivestorm), averaged mean score in $[0, 1]$ by host OS over all 16 hosts, each the mean of 3 episodes.

Model	Linux (9)	Windows (6)	FreeBSD (1)	All (16)
DeepSeek-V4-Flash	0.78	0.67	0.68	0.73
MiniMax-M3	0.80	0.68	0.50	0.74
MiniMax-M2.7	0.50	0.43	0.43	0.47
Qwen3.5-4B	0.33	0.28	0.23	0.31
Qwen3.5-9B	0.30	0.25	0.27	0.28
Qwen3.5-27B	0.48	0.41	0.38	0.45
Qwen3.5-35B	0.73	0.63	0.43	0.67
Qwen3.5-122B	0.52	0.45	0.47	0.49

6.6 Contamination: disclosure year

Every meta2 vulnerability was public by 2012, so meta2 scores alone cannot separate remediation skill from memorized fixes. Splitting sr-modern by CVE year, discovery-required pass@5 is lower on CVEs from 2023 on for all eight models, but no single difference exceeds what these scenario counts can resolve, 28 to 37 points by model. Only 64 of sr-modern's 117 scenarios carry a CVE at all (Appendix G).

6.7 Cost

The gap also costs compute. Mean working time per episode is 1.1 to 6.7 times higher without the report (median 2.7×, Table 7), and token and FLOP cost rises in every cell but Qwen3.5-35B (Appendix F).

7 Benchmark Validity

A benchmark's scores are only as trustworthy as its verifier. We audit SysRepair-Bench on three fronts (a ceiling solver, gaming adversaries, and a tamper attack on the grader), each run as a scripted agent through the *same* grading path a model uses. The gaming sweeps cover the entire benchmark.

Ceiling. A scripted reference solver that applies each suite's shipped remediation joint-passes all scenarios, so on those suites every scenario is solvable, and the grader accepts the intended fix. A negative probe alone cannot distinguish a control in place from a service that never started, so negative probes are paired with a positive control, in the PoC checks or through the regression component, and an unexecutable probe is recorded as a *failure*.

Gameability. We attack the verifier with three adversaries designed to pass a weak grader (Table 6), each swept over all scenarios: a *service-killer* that closes the vulnerability by destroying the service, a *probe-blocker* that firewalls only the exact PoC signature, and a *no-op*. All three joint-pass **nothing**.

The service-killer tests the failure mode the benchmark is designed to catch. By stopping the service, it passes the security check on 21 of the 297 scenarios (7.1%: vulnhub $7/30$, sr-modern $13/117$, ccdc $1/50$, and 0 on meta2, meta3-ubuntu, meta3-windows and ad-vm), since a stopped service cannot be exploited, yet it joint-passes none. A grader that checked only the exploit would therefore have credited about one scenario in fourteen to an agent that disabled the service; the regression component

rejects all of them. The probe-blocker, which filters only the exact PoC traffic, passes the security check on a single scenario (0.3%), so the PoC checks do not depend on one narrow signature.

A tamper attack we disclose and mitigate. Our fourth adversary attacks the grading machinery rather than the vulnerability: because the grader runs inside the sandbox, an agent can shadow its hooks so the verifier exits "success" without producing a verdict, scoring a full joint pass under a naive exit-code scorer. We harden the scorer to fail closed on that signature rather than trust the exit code.

8 Limitations and Conclusion

Limitations. Four factors bound this study. **(1) Model set.** No closed-weight model appears, so whether the strongest saturates the benchmark is open. **(2) Power.** Each cell rests on $E{=}3$ episodes per scenario and we report point estimates, so differences of a few points between models may fall within run-to-run variation. **(3) Verifier tamperability.** Linux grading runs in-sandbox; we fail closed (§7), and out-of-container grading is the universal fix. **(4) Scope.** Passing certifies the disclosed weakness is closed, and the probed service survives, not that the system is correct in every sense.

Conclusion. SysRepair-Bench evaluates a task no prior benchmark poses: closing a disclosed vulnerability on a live system while the service it protects stays available. Grading both objectives together makes remediation by service destruction measurable, and shows that locating a fault costs agents more than repairing a named one.

Reproducibility Statement

Every result in this paper is produced by one open harness built on Inspect AI (UK AI Security Institute, 2024). We release: the complete benchmark (313 scenarios as container and VM build definitions), the two-component verifiers, the scripted validity agents (the reference solver and the four gaming adversaries), and the evaluation and analysis code (pass@k extraction, and the CDR estimator). Serving stack, context window, sampling regime, and tool-call format are in Appendix A; per-episode budgets and the watchdog are specified in Appendix B. An anonymized artifact accompanies this submission.

Ethics Statement

SysRepair-Bench is a *defensive* benchmark: every task asks an agent to *remediate* a disclosed weakness on a system it must keep running, not to discover or exploit one. It nonetheless carries dual-use considerations, because each scenario ships a working vulnerable system and a proof-of-concept that exercises the flaw. We weigh these deliberately. First, every vulnerability in the corpus is already public (a disclosed CVE, a documented misconfiguration, or a well-known weak default), so the artifact discloses no new attack. The proof-of-concept code is minimal, scoped to trigger the specific flaw for grading. Second, there is direct precedent for releasing intentionally vulnerable systems for defensive research (e.g. the Metasploitable images, and the CTF and exploitation benchmarks that ship live vulnerable targets: Zhang et al., 2025b;Zhu et al., 2025; Wang et al., 2025), and the defensive-evaluation value (measuring whether autonomous agents can remediate *without* taking down the services they protect, before such agents are deployed on real infrastructure) is, in our judgment, net positive and directly relevant to informed governance of agentic AI. Third, our protocol logs rather than routes around provider safety-layer refusals: where a provider declines a defensive-remediation task, we treat that refusal as a finding about deployment-time friction for defenders rather than circumventing it. We commit to responsible disclosure for any exploitable defect the audit surfaces in third-party software, and the released harness grades without requiring the agent's exploit code to be redistributed.

AI-USE STATEMENT

The subject of this paper is the behavior of large language models, which are evaluated as described in the main text. Beyond that, the authors used LLM-based tools to assist with drafting and editing prose and with routine code scaffolding; all experimental design, benchmark construction, verifier logic, analysis, and claims were authored and verified by the authors, and every empirical number is produced by the harness and its analysis code. The authors take full responsibility for the content.

REFERENCES

Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in AI safety. *arXiv preprint arXiv:1606.06565*, 2016.

TJ Byun, Cornelius Aschermann, Kai Yuan Thng, Wu Zhou, Yupeng Yang, Lauren Deason, and Joshua Saxe. AutoPatchBench: A benchmark for AI-powered security fixes. Meta Engineering Blog; CyberSecEval 4, PurpleLlama, 2025.

Yinfang Chen, Manish Shetty, Gagan Somashekar, Minghua Ma, Yogesh Simmhan, Jonathan Mace, Chetan Bansal, Rujia Wang, and Saravan Rajmohan. AIOpsLab: A holistic framework to evaluate AI agents for enabling autonomous clouds. In *Proceedings of Machine Learning and Systems (MLSys)*, 2025. arXiv:2501.06706.

Jackson Clark, Yiming Su, Saad Mohammad Rafid Pial, Yifang Tian, Lily Gniedziejko, Hans-Arno Jacobsen, Yinfang Chen, and Tianyin Xu. SREGym: A live benchmark for AI SRE agents with high-fidelity failure scenarios. *arXiv preprint arXiv:2605.07161*, 2026.

Defense Advanced Research Projects Agency (DARPA). Artificial intelligence cyber challenge (AIxCC). DARPA Research Program, `https://www.darpa.mil/research/programs/ai-cyber`, 2023.

Stefano Fumero, Kai Huang, Matteo Boffa, Danilo Giordano, Marco Mellia, Zied Ben Houidi, and Dario Rossi. CyberSleuth: Autonomous blue-team LLM agent for web attack forensics. *arXiv preprint arXiv:2508.20643*, 2025.

Wenqi Huang, Charley Lee, Leonard Tng, and Serena Ge. DeepSWE: Measuring frontier coding agents on original, long-horizon engineering tasks, 2026. URL `https://arxiv.org/abs/2607.07946`.

Saurabh Jha, Rohan Arora, Yuji Watanabe, Takumi Yanagawa, Yinfang Chen, Jackson Clark, Bhavya Bhavya, Mudit Verma, Harshit Kumar, Hirokuni Kitahara, Noah Zheutlin, Saki Takano, Divya Pathak, Felix George, Xinbo Wu, Bekir O. Turkkan, Gerard Vanloo, Michael Nidd, Ting Dai, Oishik Chatterjee, Pranjal Gupta, Suranjana Samanta, Pooja Aggarwal, Rong Lee, Pavankumar Murali, Jae wook Ahn, Debanjana Kar, Ameet Rahane, Carlos Fonseca, Amit Paradkar, Yu Deng, Pratibha Moogi, Prateeti Mohapatra, Naoki Abe, Chandrasekhar Narayanaswami, Tianyin Xu, Lav R. Varshney, Ruchi Mahindru, Anca Sailer, Laura Shwartz, Daby Sow, Nicholas C. M. Fuller, and Ruchir Puri. ITBench: Evaluating AI agents across diverse real-world IT automation tasks, 2025. URL `https://arxiv.org/abs/2502.05352`.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. SWE-bench: Can language models resolve real-world GitHub issues?, 2024. URL `https://arxiv.org/abs/2310.06770`.

Joint Task Force. Security and privacy controls for information systems and organizations. Technical Report NIST SP 800-53 Rev. 5, National Institute of Standards and Technology, 2020.

Victoria Krakovna, Jonathan Uesato, Vladimir Mikulik, Matthew Rahtz, Tom Everitt, Ramana Kumar, Zac Kenton, Jan Leike, and Shane Legg. Specification gaming: The flip side of AI ingenuity. DeepMind blog, 2020.

Nancy Lau, Louis Sloot, Jyoutir Raj, Giuseppe Marco Boscardin, Evan Harris, Dylan Bowman, Mario Brajkovski, Jaideep Chawla, and Dan Zhao. ZeroDayBench: Evaluating LLM agents on unseen zero-day vulnerabilities for cyberdefense. In *ICLR 2026 Workshop on Agents in the Wild*, 2026. arXiv:2603.02297.

Hwiwon Lee, Ziqi Zhang, Hanxiao Lu, and Lingming Zhang. SEC-bench: Automated benchmarking of LLM agents on real-world software security tasks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. arXiv:2506.11791.

Seunghyun Lee and David Brumley. ExploitBench: A capability ladder benchmark for LLM cybersecurity agents, 2026. URL https://arxiv.org/abs/2605.14153.

Yuqiao Meng, Luoxi Tang, Feiyang Yu, Xi Li, Guanhua Yan, Ping Yang, and Zhaohan Xi. Benchmarking LLM-assisted blue teaming via standardized threat hunting. *arXiv preprint arXiv:2509.23571*, 2025.

Mike A. Merrill, Alexander G. Shaw, Nicholas Carlini, et al. Terminal-Bench: Benchmarking agents on hard, realistic tasks in command line interfaces, 2026. arXiv:2601.11868.

Daniel Shepard and Robin Salimans. AutomationBench, 2026. URL https://arxiv.org/abs/2604.18934.

Maxwell Standen, Martin Lucas, David Bowman, Toby J. Richer, Junae Kim, and Damian Marriott. CybORG: A gym for the development of autonomous cyber agents, 2021. arXiv:2108.09118; TTCP CAGE Challenge series.

Alexander Matt Turner, Neale Ratzlaff, and Prasad Tadepalli. Avoiding side effects in complex environments. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. arXiv:2006.06547.

UK AI Security Institute. Inspect AI: Framework for large language model evaluations. Software framework, https://github.com/UKGovernmentBEIS/inspect_ai, 2024.

Lehan Wang, Boli Chen, Ruixue Ding, Pengjun Xie, et al. SecRespond: Benchmarking AI agents for real-world post-compromise incident response, 2026a. arXiv:2607.26791.

Zhun Wang, Tianneng Shi, Jingxuan He, Matthew Cai, Jialin Zhang, and Dawn Song. CyberGym: Evaluating AI agents' real-world cybersecurity capabilities at scale, 2025. arXiv:2506.02548.

Zhun Wang, Nico Schiller, Hongwei Li, Srijiith Sesha Narayana, Milad Nasr, Nicholas Carlini, Xiangyu Qi, Eric Wallace, Elie Bursztein, Luca Invernizzi, Kurt Thomas, Yan Shoshitaishvili, Wenbo Guo, Jingxuan He, Thorsten Holz, and Dawn Song. ExploitGym: Can AI agents turn security vulnerabilities into real attacks?, 2026b. URL https://arxiv.org/abs/2605.11086.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, et al. OSWorld: Benchmarking multimodal agents for open-ended tasks in real computer environments. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. arXiv:2404.07972.

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

Andy K. Zhang, Joey Ji, Celeste Menders, Riya Dulepet, et al. BountyBench: Dollar impact of AI agent attackers and defenders on real-world cybersecurity systems. arXiv preprint arXiv:2505.15216, 2025a.

Andy K Zhang, Neil Perry, Riya Dulepet, Joey Ji, Celeste Menders, Justin W Lin, Eliot Jones, Gashon Hussein, Samantha Liu, Donovan Julian Jasper, Pura Peetathawatchai, Ari Glenn, Vikram Sivashankar, Daniel Zamoshchin, Leo Glikbarg, Derek Askaryar, Haoxiang Yang, Aolin Zhang, Rishi Alluri, Nathan Tran, Rinnara Sangpisit, Kenny O Oseleononmen, Dan Boneh, Daniel E. Ho, and Percy Liang. Cybench: A framework for evaluating cybersecurity capabilities and risks of language models. In *The Thirteenth International Conference on Learning Representations*, 2025b. URL https://openreview.net/forum?id=tc90LV0yRL.

Yuxuan Zhu, Antony Kellermann, Dylan Bowman, Philip Li, Akul Gupta, Adarsh Danda, Richard Fang, Conner Jensen, Eric Ihli, Jason Benn, Jet Geronimo, Avi Dhir, Sudhit Rao, Kaicheng Yu, Twm Stone, and Daniel Kang. CVE-Bench: A benchmark for AI agents' ability to exploit real-world web application vulnerabilities. In *Proceedings of the 42nd International Conference on Machine Learning*, ICML'25. JMLR.org, 2025.

A MODEL CONFIGURATIONS

Panel A (cross-lab hosted). DeepSeek-V4-Flash, MiniMax-M3 and MiniMax-M2.7 are queried through their hosted OpenAI-compatible APIs under the ReAct scaffold.

Panel B (within-family Qwen3.5 ladder). Served locally under vLLM at each model's full 262,144-token context window, in the Qwen3.5 XML tool-call format.

Parameter counts. The models span 4B to 428B total parameters. Panel B (Qwen3.5, Apache-2.0): 4B, 9B, and 27B dense; 35B-A3B and 122B-A10B mixture-of-experts (total / active). Panel A: DeepSeek-V4-Flash (284B / 13B active) and MiniMax-M3 (428B / 23B) and MiniMax-M2.7 (230B / 10B). All are open-weight; the FLOP column of Table 7 uses active parameters.

Sampling. Multi-attempt pass@k is only meaningful under stochastic decoding; we use each provider's / vLLM's default (non-greedy) sampling and do not force greedy decoding for the multi-attempt conditions. The locally served Panel B models are pinned by checkpoint. The hosted Panel A APIs cannot be pinned by revision, so we record the endpoint and access date and treat those results as reproducible only up to provider-side changes.

B EPISODE BUDGETS AND WATCHDOG

Each episode is bounded by a wall-clock `time_limit` of 3600 s and a working-time budget (`working_limit`) of 2400 s; the per-episode message and token caps are disabled, so time is the binding budget. The agent is given k=5 submit attempts within one episode. We run E=3 independent episodes per (model, scenario, condition).

C QWEN3.5 LADDER DETAIL

Table 4: Within-family Qwen3.5 ladder (§6.4, Figure 3). pass@5 is the benchmark-wide value of Table 8.

	Report-informed		Discovery-required	
Rung	pass@5	CDR (%)	pass@5	CDR (%)
Qwen3.5-4B	72.2	7.2	24.7	18.4
Qwen3.5-9B	70.9	4.0	29.8	9.6
Qwen3.5-27B	80.3	5.1	41.4	8.6
Qwen3.5-35B	80.8	4.2	68.2	1.4
Qwen3.5-122B	87.8	4.0	45.4	7.9

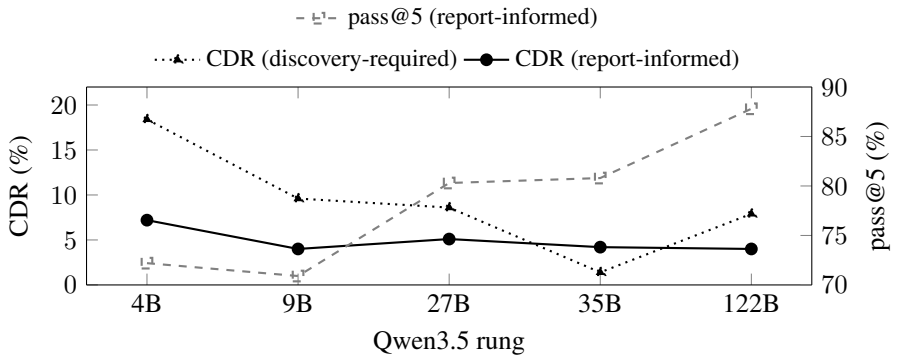

Figure 3: Within-family Qwen3.5 ladder. Report-informed pass@5 (right axis); report-informed and discovery-required CDR (left axis). Per-rung values are in Table 4.

D ADDITIONAL RESULTS TABLE

Table 5: Composition of all 313 scenarios by suite, runtime and remediation category. Categories are pinned per scenario; each Hivestorm host mixes several, so it has none. Runtime: C = Linux container, K = kernel VM, W = Windows container, V = Windows VM, B = FreeBSD VM.

Suite	Runtime	Config. Hard.	Access Ctrl.	Network Sec.	Comp. Controls	Dependency	Total
meta2	C	15	3	11	7	4	40
sr-modern	C (113), K (4)	27	26	18	24	22	117
vulnhub	C	16	9	3	0	2	30
ccdc	C	9	21	18	0	2	50
meta3-ubuntu	C	5	2	1	7	4	19
meta3-windows	W	8	2	9	2	0	21
ad-vm	V	5	5	1	8	1	20
Binary total		**85**	**68**	**61**	**48**	**35**	**297**
hivestorm	C (9), W (5), V (1), B (1)			mixed per host			16
Total							**313**

Table 6: Verifier gaming audit. Scripted adversaries run through the same grading path as a model. Security = share of scenarios where the exploit is blocked; JOINT = share passing both the security and regression components.

Adversary	security	**JOINT**
No-op (declare done)	0.3%	**0.0%**
Service-killer	7.1%	**0.0%**
Probe-blocker (firewall the PoC signature)	0.3%	**0.0%**
Verifier-tamperer (attack the grader)	–	**0.0%**

Scenario (meta2-08). A VNC server uses the weak password `"password"`; the availability requirement is that remote desktop access stays reachable. PoC: weak password must be rejected. Regression: `x11vnc` running, listening on `:5900`, and authenticating.

Joint pass (DeepSeek-V4-Flash)	**Collateral damage** (Qwen3.5-4B)
`x11vnc -storepasswd <strong>` `systemctl restart x11vnc`	`pkill x11vnc` *("removes" the weak-password exposure)*
s: weak password rejected ✓ r: VNC up, :5900, auth ✓ **joint = pass**	s: weak password rejected ✓ r: `x11vnc` not running ✗ **joint = fail (collateral)**

Figure 4: Worked example of the two-component verdict on meta2-08: a joint pass (left) and a collateral-damage failure (right). s = security check, r = regression check.

Table 7: Mean cost per episode benchmark-wide, report-informed (RI) vs. discovery-required (DR), $\pm$ standard error over episodes. ktok = input plus output tokens; PFLOP $= 2\,N_{\text{active}}(\text{in} + \text{out})$, so it prices compute rather than billing. s = working-time seconds.

	Report-informed			Discovery-required		
Model	ktok	PFLOP	s	ktok	PFLOP	s
DeepSeek-V4-Flash	37 ± 2	0.96 ± 0.04	224 ± 10	131 ± 3	3.40 ± 0.07	636 ± 15
MiniMax-M3	47 ± 3	2.16 ± 0.14	236 ± 9	382 ± 16	17.6 ± 0.7	867 ± 27
MiniMax-M2.7	47 ± 2	0.94 ± 0.04	354 ± 10	267 ± 7	5.3 ± 0.1	802 ± 24
Qwen3.5-4B	1693 ± 136	13.5 ± 1.1	391 ± 17	6565 ± 153	52.5 ± 1.2	978 ± 14
Qwen3.5-9B	680 ± 30	12.2 ± 0.5	405 ± 10	3741 ± 90	67.3 ± 1.6	1263 ± 18
Qwen3.5-27B	254 ± 16	13.7 ± 0.8	888 ± 30	1017 ± 49	54.9 ± 2.6	1500 ± 31
Qwen3.5-35B	1388 ± 218	8.3 ± 1.3	352 ± 17	590 ± 17	3.5 ± 0.1	387 ± 16
Qwen3.5-122B	269 ± 18	5.4 ± 0.4	333 ± 17	638 ± 22	12.8 ± 0.4	2224 ± 42

Table 8: Joint pass@1 and pass@5 benchmark-wide by model and condition. S = scenarios; Macro = unweighted mean of the five unrounded per-suite pass@5 rates of Table 2; best pass@5 per condition in bold.

Model	Cond.	S	@1	@5	Macro
DeepSeek-V4-Flash	report	297	86.3	**95.4**	96.5
MiniMax-M3	report	297	80.4	89.9	92.5
MiniMax-M2.7	report	297	74.5	83.1	86.6
Qwen3.5-4B	report	297	63.9	72.2	75.9
Qwen3.5-9B	report	297	62.9	70.9	75.1
Qwen3.5-27B	report	297	75.3	80.3	85.0
Qwen3.5-35B	report	297	71.9	80.8	85.7
Qwen3.5-122B	report	297	79.3	87.8	90.4
DeepSeek-V4-Flash	discovery-required	297	55.6	**73.0**	70.5
MiniMax-M3	discovery-required	297	46.8	58.5	55.2
MiniMax-M2.7	discovery-required	297	35.8	47.2	46.0
Qwen3.5-4B	discovery-required	297	18.9	24.7	28.4
Qwen3.5-9B	discovery-required	297	22.9	29.8	31.3
Qwen3.5-27B	discovery-required	297	36.2	41.4	39.9
Qwen3.5-35B	discovery-required	297	57.1	68.2	64.7
Qwen3.5-122B	discovery-required	297	37.8	45.4	44.0

E AGGREGATION

Truncated episodes in pass@k. pass@k for an episode is 1 iff any of its first k attempts joint-passes. Success is *absorbing*: the harness stops at the first passing attempt, so a solved episode records fewer than k attempts and still counts as a pass at every larger k.

An episode that never passes and is observed for fewer than k attempts has exhausted its harness budget (time or working limit) without repairing the host. We score such episodes as **failures** at every k. Excluding them instead would not be neutral: truncation is correlated with failure, because an agent that flounders consumes its budget while one that succeeds stops early, so excluding truncated episodes removes failures from the denominator and biases pass@k upward. Scoring them as failures also holds the denominator fixed across k, so no part of the curve is computed on a smaller set of scenarios than any other. This is the conservative choice and it matches the protocol as run: a host still unrepaired when the budget expires is a non-solve.

The micro-average of §5 is $\frac{1}{N}\sum_i \bar{j}_i$ over all N scenarios; the macro-average is the Macro column of Table 8.

F COST DETAIL

The ratio of discovery-required to report-informed mean working time per model (Table 7) is 2.84 for DeepSeek-V4-Flash, 3.67 for MiniMax-M3, 2.27 for MiniMax-M2.7, 2.50, 3.12, 1.69, 1.10 and 6.68 for the Qwen3.5 rungs from 4B to 122B, a median of 2.67. The free-roam track is excluded.

G CONTAMINATION DETAIL

Table 9 splits the 59 dated container-track sr-modern scenarios (the 64 with a CVE, less the four kernel-level ones and one not-applicable) by the year in their CVE identifier: 33 before 2023 and 26 from 2023 on. Discovery-required pass@5 is lower on the later CVEs for all eight models, by 0.2 to 27.0 points. No single difference exceeds the minimum these scenario counts detect at 80%

power and two-sided α=0.05, which is 28 to 37 points at each model's observed pass rate. The arms are not severity-matched (22 of 33 earlier CVEs are Critical against 14 of 26 later ones), which is an alternative explanation for any difference.

Table 9: sr-modern pass@5 by CVE year. Pre = CVE year before 2023, post = 2023 or later.

	Report-informed		Discovery-required	
Model	pre	post	pre	post
DeepSeek-V4-Flash	100.0	97.4	87.9	71.2
MiniMax-M3	94.9	93.6	66.7	65.4
MiniMax-M2.7	85.5	82.7	57.8	30.8
Qwen3.5-4B	68.7	69.2	18.2	17.9
Qwen3.5-9B	64.1	69.2	24.2	16.7
Qwen3.5-27B	84.8	78.2	46.5	20.5
Qwen3.5-35B	84.8	83.3	78.8	70.5
Qwen3.5-122B	85.9	84.6	51.5	30.8

The binding constraint is coverage, and it is structural. Only 64 of sr-modern's 117 scenarios carry a CVE identifier and can be placed on a disclosure timeline at all. We audited the remaining 53 and none of them can be dated, because they are not disclosed defects: they are configuration states (an unauthenticated daemon, a mounted container socket, a public object-store policy), protocol-level design properties, or deliberately-vulnerable teaching applications. Such scenarios have no disclosure event to date. This bounds what any temporal split on this suite can establish, and it is a property of what system-repair scenarios are made of rather than a gap in the annotation: a benchmark of realistic misconfigurations will always contain many items that no disclosure timeline can order.

H FAILURE TAXONOMY DETAIL

Table 10 gives the composition. *Discovery-required* failures are dominated by non-remediation (72.0–87.7% across models). *Report-informed* failures involve collateral damage (alone or with non-remediation) far more often: 24.8–45.1% across models, against 12.3–28.0% for discovery-required failures.

Table 10: Failure taxonomy benchmark-wide (% of failed episodes; "Other" = 0% by construction, so rows sum to 100). n = failed episodes whose verifier reported both components (the pool CDR is computed over). DR = discovery-required, RI = report-informed.

Model	Cond.	n	Collateral damage	Not remediated	Both
DeepSeek-V4-Flash	RI	27	18.5	63.0	18.5
DeepSeek-V4-Flash	DR	212	6.6	82.1	11.3
MiniMax-M3	RI	51	31.4	54.9	13.7
MiniMax-M3	DR	144	8.3	81.9	9.7
MiniMax-M2.7	RI	26	30.8	61.5	7.7
MiniMax-M2.7	DR	118	14.4	72.0	13.6
Qwen3.5-4B	RI	192	30.2	58.9	10.9
Qwen3.5-4B	DR	487	7.8	81.7	10.5
Qwen3.5-9B	RI	157	14.0	75.2	10.8
Qwen3.5-9B	DR	456	5.3	85.5	9.2
Qwen3.5-27B	RI	108	29.6	60.2	10.2
Qwen3.5-27B	DR	373	7.8	83.4	8.8
Qwen3.5-35B	RI	99	26.3	64.6	9.1
Qwen3.5-35B	DR	126	4.0	84.9	11.1
Qwen3.5-122B	RI	78	33.3	57.7	9.0
Qwen3.5-122B	DR	163	6.1	87.7	6.1