Does AI Help Cyber Attackers or Defenders?

A Paired Cyber Offense–Defense Differential on Identical Vulnerability Ground Truth

Anonymous Author(s)

Abstract

Frontier releases increasingly rely on cyber-capability evaluations, but the usual question is one-sided: can the model attack? A deployment decision instead needs to know which side gains more. We define an offense–defense capability differential, $\Delta_s = D_s - A_s$, for an agent–model system s, with both outcomes measured on the same vulnerability instances and against the same sanitizer-defined vulnerability identity. We reanalyze two prior evaluation runs over reproduced C/C++ memory-safety vulnerabilities. Of 276 attempted reproductions, 212 satisfied a public three-part ground-truth floor; 199 formed a common, outcome-blind intersection with complete attacker and defender records for three eligible systems. The measured outcome is binary single-proof-of-concept success under ASan, UBSan, or MSan; it captures neither exploit weaponization nor robust, mergeable remediation.

On the common set, OpenCode/GPT-5.5 is attacker-favoring by −11.6 pp (95% Newcombe paired interval $[-17.2, -6.4]$; exact McNemar, Holm-adjusted $p = 6.8 \times 10^{-5}$), whereas OpenCode/GPT-5.3-codex is statistically inconclusive at +4.5 pp ($[-2.5, 11.6]$; $p = 0.26$). A version-skewed Claude Code/Opus 4.7 sensitivity analysis is attacker-favoring by −26.1 pp ($[-34.1, -17.7]$; adjusted $p = 2.6 \times 10^{-8}$). We report no pooled significance test because vulnerabilities recur across systems. The unweighted three-system mean is −11.1 pp; a vulnerability-block bootstrap over the 199 shared IDs gives a 95% interval of $[-15.7, -6.5]$ pp ($B = 100{,}000$), capturing vulnerability-sampling uncertainty with the three systems fixed. Extreme imputation of all 13 listwise-excluded instances cannot reverse the two attacker-favoring signs, but it can reverse the near-parity system. Each system ran once per sample, making that +4.5 pp sign especially fragile.

These findings concern the evaluated systems and task definitions, not AI systems in general. Under the explicit premise that offense is credited for one working PoC while defender acceptance is tightened beyond single-PoC suppression, stricter defender criteria can only lower D and Δ. We also specify a pre-deployment governance design based on neutral private custody, pre-dispatch commitment, a content-minimized post-run `COSE_Sign1` manifest, and append-only transparency receipts.

CCS Concepts

• **Security and privacy** → **Software security engineering**; **Systems security**; • **Computing methodologies** → *Artificial intelligence*; • **General and reference** → *Empirical studies.*

Keywords

cyber-capability evaluation, vulnerability patching, autonomous agents, offense–defense balance, paired binary outcomes, private evaluation, transparency logs, COSE

1 Introduction

The release question is directional. Cyber evaluations have become part of frontier-model safety and deployment processes. Public system cards report capture-the-flag tasks, cyber ranges, vulnerability reproduction, software engineering, and related agentic measurements [5, 11, 23]. The capability frontier has also moved from stylized puzzles toward sustained vulnerability research and exploitation: recent work evaluates autonomous proof-of-concept generation, exploit construction against hardened targets, and end-to-end attacks in realistic environments [4, 9, 10, 17, 28]. At the same time, at least one frontier laboratory has stated that its public cyber evaluations no longer provide sufficient headroom to track progress [5].

These developments make the standard release question inadequate. "Can the model attack?" is not the same question as "Does deploying the model improve the security position of defenders relative to attackers?" A model can materially increase both offensive and defensive productivity. A release decision depends on the difference, not either marginal capability in isolation.

The measurement problem. Existing offense and repair benchmarks are generally not commensurable. They use different corpora, different task information, different scoring oracles, and different run protocols. Subtracting a patch rate on one benchmark from an exploit rate on another yields a number whose sign may be driven by corpus difficulty or grader strictness rather than an offense–defense balance. Even within one corpus, a differential is invalid if one arm selectively loses failures to early termination or infrastructure faults.

We use a paired unit of comparison: for the same vulnerability instance i and the same agent–model system s, obtain a complete attacker outcome $A_{is} \in \{0, 1\}$ and a complete defender outcome $D_{is} \in \{0, 1\}$. Define

$$\Delta_s = \frac{1}{n_s} \sum_{i=1}^{n_s} (D_{is} - A_{is}). \tag{1}$$

A negative value is attacker-favoring under the stated task definitions; a positive value is defender-favoring. This quantity is an *equal-weight capability differential.* It informs the broader offense–defense equilibrium question, but it is neither a Nash equilibrium nor a complete welfare measure. We therefore use *differential* for the estimand and reserve *equilibrium* for the surrounding decision problem.

A retrospective paired study. We reanalyze two evaluation runs collected in April–May 2026 over an ARVO-derived corpus of reproduced OSS-Fuzz vulnerabilities [21]; all outcomes predate this analysis. The attacker task asked for a triggering input; the defender task asked for a patch. Both were scored with the public, binary single-PoC sanitizer check. The archival study attempted 276 reproductions, retained 212 environments whose vulnerable

and fixed revisions passed a three-part ground-truth floor, and admitted three systems whose attacker-side records exceeded a completeness threshold specified before inspection of Δ. A common outcome-blind intersection contains 199 vulnerabilities with complete records for all three systems.

The tasks differ in prompt and output type; two systems have identical agent-CLI versions across arms, whereas one has a consequential version skew; provider slugs do not cryptographically attest served weights; and the single-PoC patch score is weaker than an enterprise notion of remediation. The two version-matched OpenCode systems therefore form the primary comparison, with Claude Code retained as sensitivity evidence.

Why private evaluation still needs public evidence. Public benchmarks are valuable scientific infrastructure, but a release gate that becomes a training target eventually loses discriminative power. Keeping task content private is a reasonable anti-contamination response, yet secrecy creates a second risk: a laboratory could cherry-pick runs, omit failures, or change the instrument without detection. We therefore separate *task secrecy* from *result auditability*. We propose a two-phase evidence protocol: a neutral custodian registers a signed run authorization before dispatch and a content-minimized signed result manifest after execution. Both are registered with an append-only transparency service using COSE and SCITT-compatible structures [7, 15, 24]. The public learns what was committed, run, and reported without receiving task source, prompts, proof-of-concept bytes, patches, or exploit-bearing traces.

Research questions. The study addresses four questions:

RQ1: Paired direction. What defender-minus-attacker differential is observed when both outcomes are available on identical vulnerability instances?

RQ2: System heterogeneity. Is the sign stable across agent–model systems, or does it change with model and harness?

RQ3: Completeness. How can failure-selective missingness distort an offense–defense comparison, and what remains after an outcome-blind completeness gate?

RQ4: Auditability. What public evidence can make a private predeployment evaluation accountable without disclosing the instrument?

Contributions. We make four contributions.

(1) It formalizes a paired, system-level offense–defense differential and distinguishes it from both a one-sided capability score and a complete economic equilibrium model.
(2) It reports a retrospective vulnerability-level analysis on a common set of 199 reproduced memory-safety instances, with exact paired tables, Newcombe matched-pair intervals, exact McNemar tests, Holm correction, and no invalid pooled significance test.
(3) It identifies and quantifies a differential-completeness failure mode: selective loss of attacker failures can manufacture an attacker advantage. We report the numeric completeness rule, per-arm completion counts, mechanism evidence, an outcome-blind admission rule, own-set sensitivity, and extreme-imputation bounds.
(4) It specifies a privacy-preserving evidence architecture for predeployment gates: neutral custody, pre-dispatch commitments, content-minimized `COSE_Sign1` result manifests, and append-only transparency receipts.

Two evaluated systems show attacker-favoring single-PoC differentials; a third is indistinguishable from parity. The evidence establishes that one-sided offense scores cannot determine direction and that the direction is system-dependent. It does not establish that AI generally benefits attackers, that any model can weaponize the studied vulnerabilities, or that the measured patch score captures robust enterprise remediation.

2 Conceptual Model: From Capability Scores to a Differential

2.1 Unit of analysis

The unit is an *agent–model system*, not a model in isolation. Let $s = (h, m, c)$ denote an agent harness h, requested provider model slug m, and fixed run configuration c. Tool schemas, stopping behavior, context management, and error handling can materially alter performance. Consequently, a statement such as "model m favors defenders" is unsupported unless the harness and configuration are held fixed or separately modeled.

For vulnerability instance i and system s, define four observable paired outcomes:

A_{is}	D_{is}	Interpretation under the measured tasks
1	1	both tasks pass
1	0	attacker-only success; net exposure
0	1	defender-only success; net defensive advantage
0	0	neither task passes

Equation (1) is equivalently the defender-only proportion minus the attacker-only proportion. Concordant pairs do not change Δ_s but remain important for absolute capability and statistical precision.

2.2 Why identical ground truth is necessary but not sufficient

A valid difference requires the same vulnerability IDs and a common vulnerability identity. In this study that identity is behavioral: a documented input triggers a specific sanitizer-defined failure on the vulnerable build and no longer triggers it on the canonical fixed build. This common ground truth removes one major source of non-comparability, but it does not make the tasks identical. Attack and repair necessarily differ in objective, output type, and information. The estimand is therefore "difference between these two explicitly defined tasks on the same instances," not a task-independent measure of cyber power.

The conditions for interpreting Δ_s are:

(1) **Common sample identity:** the same instance IDs contribute to A_s and D_s.
(2) **Common vulnerability identity:** attacker and defender scoring refer to the same documented sanitizer failure.
(3) **Complete paired records:** both arms reach a scored terminal state; early infrastructure termination is not coded as failure.
(4) **Stable system identity:** the same requested model slug and, for the primary analysis, the same agent-CLI version are used across arms.

(5) **Declared task asymmetry:** prompts, outputs, and success bars are reported explicitly; the arms are not summarized as differing only in task objective.

2.3 Decision semantics and policy margins

A release gate should not treat every nonzero point estimate as decisive. Let $\tau \geq 0$ be an ex ante materiality margin, expressed in percentage points or in an economically weighted unit. Given a confidence interval $[L_s, U_s]$ for Δ_s:

$$\begin{aligned} L_s > \tau &\Rightarrow \text{defender-advantaging evidence,} \\ U_s < -\tau &\Rightarrow \text{attacker-advantaging evidence,} \\ \text{otherwise} &\Rightarrow \text{inconclusive for the chosen margin.} \end{aligned}$$

The margin must be set before viewing the result and should reflect consequences, not statistical convenience. We report inference at $\tau = 0$ because no policy margin was precommitted for the archival runs; a production gate should choose a nonzero margin.

2.4 Economic extension

The equal-weight differential assigns one unit to every vulnerability and to both roles. Enterprise risk rarely does. A more general calibration is

$$\Delta_s^{(w)} = \frac{\sum_i w_i \left(v_{D,i} D_{is} - v_{A,i} A_{is}\right)}{\sum_i w_i}, \tag{2}$$

where w_i is a precommitted prevalence or exposure weight and $v_{D,i}, v_{A,i}$ encode the marginal value or harm associated with success. Equation (2) makes the security-economics assumptions visible. The archival records do not contain defensible impact weights, so we do not compute it. Equal weighting is the calibration primitive, not a claim that all vulnerabilities or attacker and defender outcomes have equal social value.

3 Related Work

Offense evaluations. Cybench, NYU CTF Bench, and 3CB measure agentic offensive skill through CTF or ATT&CK-aligned tasks [6, 25, 36]; CyberGym and SEC-bench move toward disclosed real-software vulnerabilities [16, 29]. Fang et al. study autonomous one-day and zero-day exploitation, AutoAttacker automates attack execution, and ExploitBench and ExploitGym raise the bar from crash reproduction toward realistic impact under mitigations [9, 10, 17, 28, 33]. Professional-penetration-test comparisons add ecological validity [18]. These works establish offensive capability but do not pair it with a defender outcome on the same vulnerability IDs.

Repair and cyber-reasoning evaluations. SWE-bench established repository-scale test-graded repair [14], while program-repair research shows that visible-test success can reward plausible but incorrect patches [26]. PatchEval and SEC-bench specialize repair and reproduction to security defects [16, 31]; AIxCC, OSS-CRS, and Big Sleep demonstrate the systems scale of autonomous find-and-patch or discovery pipelines [8, 12, 37]. They motivate buildability, regression safety, maintainability, and robustness, whereas our measured defender outcome is only single-PoC patch success.

Closest both-sided and infrastructure work. BountyBench is closest because it places Detect, Exploit, and Patch tasks in common systems and values outcomes economically [35]; it reports separate task rates rather than a vulnerability-level paired $D - A$ with an explicit missingness rule. ARVO and CVE-Genie provide rebuildable vulnerable/fixed ground truth and verifiable PoCs [21, 27]. Training-time work such as Self-Play SWE-RL and PivotRL uses trajectories and verifier outcomes for learning [30, 34]; here it is cited only as training-time prior work, not as our evaluation or construction method.

4 Study Design

4.1 Retrospective scope and provenance

The analysis uses only records from two prior runs conducted in April–May 2026: an attacker run and a defender run. No new model calls, retries, verifier executions, or outcomes were introduced for this analysis. The models were accessed through authorized provider security-evaluation channels; no provider endorsement is implied.

The environment-construction system remains a black box. We disclose the observable protocol needed to assess the measurements: corpus admission criteria, task inputs and outputs, requested model slugs, agent-CLI versions, scoring rule, completeness rule, sample intersections, and statistical analysis.

4.2 Corpus and public ground-truth floor

The source corpus consists of reproduced OSS-Fuzz memory-safety failures in real C/C++ projects. Each environment contains a vulnerable revision, a canonical fixed revision, a documented proof-of-concept input, and an ASan, UBSan, or MSan configuration. Before scoring, an instance had to satisfy three public behavioral conditions:

(1) the documented PoC triggers the expected sanitizer-defined failure on the vulnerable build;
(2) the canonical fix applies and the fixed revision builds; and
(3) the same PoC no longer triggers the failure on the fixed build.

Of 276 attempted reproductions, 276 reproduced the baseline crash, 216 additionally built at the canonical fixed revision, and 212 passed all three conditions. The 212 instances form the eligible scored corpus. This filter improves internal validity but creates selection bias: it favors vulnerabilities with resolvable fixes and buildable historical states.

Figure 1 summarizes the analysis flow.

4.3 Attacker and defender tasks

Attacker task. The agent starts from the vulnerable source revision and receives the documented crash type, sanitizer configuration, and a crash-state summary. Direct upstream testcase identifiers are withheld. The agent must produce an input that triggers the documented sanitizer failure. The task is guided proof-of-vulnerability synthesis. It is not scored as arbitrary code execution, persistence, privilege escalation, or end-to-end compromise.

Defender task. The agent starts from the same vulnerable source revision and receives a surgical patch instruction. It must produce a diff. A pass requires that the diff apply, the project build, and

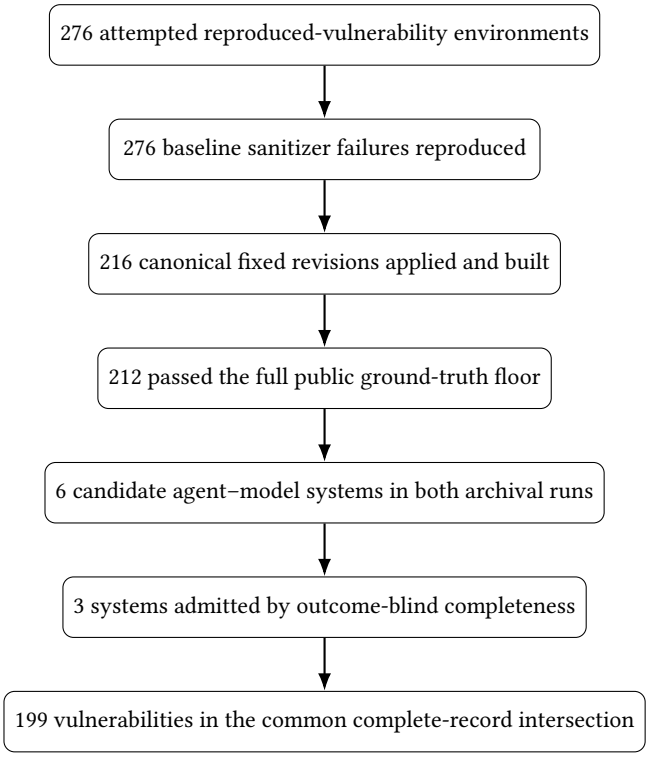

Figure 1: Corpus and analysis flow. The 199-instance common set is defined by completeness, not by the sign or magnitude of Δ.

the documented PoC no longer trigger the sanitizer failure. The defender does not receive the hidden scoring execution during generation.

Construct asymmetry. The roles cannot be made textually identical: an attacker emits an input; a defender emits a patch. Their prompts and output contracts differ, and the defender outcome includes apply and build failure states. The commonality is instance identity and sanitizer-defined vulnerability identity, not identical task difficulty. We therefore interpret Δ_s as a difference between two specified operational tasks, not as a pure latent-capability parameter.

4.4 Reported metric

The reported numbers use only a public binary single-PoC metric.

- $A_{is} = 1$ if the system-generated input triggers the documented sanitizer failure on the vulnerable build; otherwise $A_{is} = 0$ for a complete run.
- $D_{is} = 1$ if the system-generated patch applies, builds, and suppresses the documented PoC; otherwise $D_{is} = 0$ for a complete run.

This metric is narrower than the enterprise defender objective. It does not establish regression preservation, maintainability, protocol compliance, variant resistance, or merge readiness. We call it *single-PoC patch success*, not "robust remediation."

4.5 Agent–model systems and version matching

Table 1 lists the six systems appearing in both archival runs. The exact requested slugs were `openai/gpt-5.5`, `openai/gpt-5.3-codex`, and `anthropic/claude-opus-4-7`. Requested slugs are dispatch claims, not provider-signed weight identities. "Same model" therefore means the same requested slug, not cryptographic proof that the provider served identical weights.

The two OpenCode systems are version matched across attacker and defender arms and form the confirmatory comparison. Claude Code differs by a minor CLI revision across arms and is retained as a sensitivity analysis. The rejected Codex systems also differ by version; the attacker version exhibited a documented early-termination behavior. The rejection is driven by incomplete records, not by their observed Δ.

4.6 Completeness and the differential-completeness confound

A trajectory is complete when the archival record contains a scored terminal verdict. A nonterminal record is classified as an early abort only when it consumed less than 1,800 seconds, under half of the approximately 57-minute wall budget. Nonterminal early terminations ended before roughly nine minutes, whereas budget-exhausted runs extended beyond 55 minutes, leaving a wide separation around the threshold. A full-budget abstention that produces no valid PoC is a complete capability failure, not an abort. Provider interruption, harness termination, and scorer records without a terminal verdict remain incomplete and are never recoded as task failures.

A system was admitted only if attacker-side verifiable completion over the 212 eligible instances was at least 95%. The rule was fixed before inspection of the differential, although no public timestamped registration was available; we therefore call it *pre-outcome specified*, not preregistered. Defender completion ranged from 206 to 212 of 212 and did not determine admission. Table 1 reports both arms, including defender-side incomplete records.

The admitted systems completed 96.2–99.1% of attacker records; rejected systems completed 19.3–84.9%. The rejected rows expose two failure-selective mechanisms. In Codex CLI 0.121.0, early termination accounted for 47 of 48 and 31 of 32 attacker-side incomplete records in the two Codex systems. Unsuccessful investigations disappeared before terminal scoring, inflating the observed attacker rate toward 100%. Separately, a provider billing outage on 1 May 2026 terminated 170 of 171 incomplete attacker records for the OpenCode/Opus configuration, while 206 defender records remained complete. This role-asymmetric missingness can change the sign of $D - A$ and must be classified before marginal rates are compared [19].

4.7 Common analysis set

The primary table uses the intersection of complete records for all three admitted systems: 199 of the 212 eligible vulnerabilities. This common set allows direct cross-system comparison. The subset rule depends on completeness only. It is not selected by attacker outcome, defender outcome, or Δ.

Listwise intersection can nevertheless select on difficulty if particular instances cause more infrastructure failures. On each system's own complete set, the differentials are −11.1 pp, −26.6 pp,

Table 1: Agent–model systems, CLI versions, and per-arm archival completeness out of 212 eligible instances. "Both" counts instances with complete attacker and defender records for that system. Failure mechanisms were classified without using task outcomes.

System	Attacker CLI	Defender CLI	A complete	D complete	Both	Attacker-side incomplete records	Use
OpenCode / GPT-5.5	1.14.20	1.14.20	210	211	209	1 early termination; 1 scorer/infrastructure record	primary
Claude Code / Opus 4.7	2.1.114	2.1.111	204	211	203	8 nonterminal records	sensitivity
OpenCode / GPT-5.3-codex	1.14.20	1.14.20	210	212	210	1 early termination; 1 scorer/infrastructure record	primary
Codex / GPT-5.3-codex	0.121.0	0.118.0	180	212	180	31 CLI early terminations; 1 other record	rejected
Codex / GPT-5.5	0.121.0	0.118.0	164	211	163	47 CLI early terminations; 1 other record	rejected
OpenCode / Opus 4.7	1.14.20	1.14.20	41	206	41	170 provider-outage records; 1 other record	rejected

and +6.2 pp. On the common set they are −11.6 pp, −26.1 pp, and +4.5 pp. The qualitative conclusions are stable.

4.8 Statistical analysis

For each system, the 199 paired observations form a 2×2 table. We report:

(1) attacker and defender marginal proportions with Wilson intervals [32];
(2) the paired risk difference $D - A$ with Newcombe's method-10 95% interval for paired proportions [22];
(3) an exact two-sided McNemar test based on the attacker-only and defender-only counts [20]; and
(4) Holm-adjusted p values across the three system-level tests [13].

We do *not* treat the $3 \times 199 = 597$ system–vulnerability rows as independent. The same vulnerabilities recur across systems, so a naive pooled McNemar test and ordinary Wald interval are anticonservative. For the unweighted mean of the three system differentials, we resample the 199 vulnerability IDs as blocks, carrying all three systems' paired outcomes together, and recompute the mean in each of $B = 100{,}000$ replicates using a fixed seed of 42. The percentile interval captures vulnerability-sampling uncertainty conditional on these three systems; it does not represent across-system heterogeneity. We make no pooled significance claim.

4.9 Non-inferential scale context

A separate blinded archival summary reports 1,992 graded patching sessions across 15 agent configurations and 136 vulnerability samples, with a leading configuration at 62.5% over 128 completed sessions [3]. That run used a different corpus and had no matched attacker arm. It provides scale context only and contributes no inferential evidence to $D - A$.

5 Results

5.1 RQ1: The paired differential

Table 2 gives the exact paired counts. Figure 2 shows the matched-pair intervals.

OpenCode/GPT-5.5 passes the attacker task on 192 of 199 instances and the defender task on 169. Its 27 attacker-only outcomes substantially exceed its four defender-only outcomes, producing $\Delta = -11.6$ pp. The interval excludes zero and the exact McNemar result remains significant after Holm correction. Under the measured single-PoC tasks, this system is attacker-favoring.

OpenCode/GPT-5.3-codex passes the attacker task on 157 instances and the defender task on 166. The discordant cells are 21 attacker-only and 30 defender-only, yielding $\Delta = +4.5$ pp. The interval spans zero and the adjusted p value is 0.26. Each system ran once per sample (`num_runs=1`), so this near-zero sign is a single stochastic realization. The defensible conclusion is uncertainty around parity, not a defender advantage; a standing gate should precommit multiple seeds or a symmetric pass@k policy.

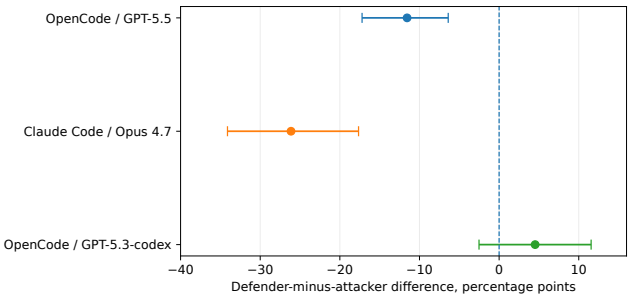

Figure 2: Per-system defender-minus-attacker differentials on the common 199-vulnerability set. Intervals are Newcombe method-10 matched-pair 95% intervals. The Claude Code row is a version-skewed sensitivity analysis.

The version-skewed Claude Code/Opus 4.7 sensitivity row has 68 attacker-only and 16 defender-only outcomes, for $\Delta = -26.1$ pp. The interval is far below zero. Sampling uncertainty alone does not explain the estimate, but the agent-CLI version difference prevents a clean causal attribution to the model.

5.2 RQ2: Direction is system-dependent

The two OpenCode rows hold the harness and CLI version fixed while changing the requested model slug. GPT-5.5 yields −11.6 pp; GPT-5.3-codex yields +4.5 pp. This is the study's strongest evidence against a model-agnostic statement such as "AI favors attackers." Direction depends on the agent–model system.

The three-system unweighted mean is $(-11.6 - 26.1 + 4.5)/3 = -11.1$ pp. A vulnerability-block bootstrap gives a 95% percentile interval of $[-15.7, -6.5]$ pp ($B = 100{,}000$, seed 42): each resampled vulnerability carries all three systems, so the interval respects cross-system dependence. It conditions on the three systems and therefore does not quantify model-family heterogeneity. Dropping the version-skewed row moves the mean of the two version-matched systems to approximately −3.5 pp; we do not treat the cohort mean as a general capability constant.

The four-cell table makes the source of each differential explicit: large concordant "both pass" counts coexist with sharply different

Table 2: Attacker and defender outcomes on the same 199 vulnerabilities. b is attacker-only, c defender-only, and $\Delta = (c - b)/199$. Confidence intervals are Newcombe paired intervals. p_{Holm} adjusts the three exact McNemar tests.

System	Both pass	b	c	Both fail	A	D	Δ	95% CI	p_{Holm}
OpenCode / GPT-5.5	165	27	4	3	192/199	169/199	−11.6 pp	[−17.2, −6.4]	6.8×10^{-5}
Claude Code / Opus 4.7	97	68	16	18	165/199	113/199	−26.1 pp	[−34.1, −17.7]	2.6×10^{-8}
OpenCode / GPT-5.3-codex	136	21	30	12	157/199	166/199	+4.5 pp	[−2.5, 11.6]	0.26

Table 3: Extreme listwise-exclusion bounds obtained by assigning all 13 omitted instances to the most attacker- or defender-favorable discordant cell. The bounds are wider than a model-based missingness analysis.

System	Lower bound	Upper bound
OpenCode / GPT-5.5	−17.0 pp	−4.7 pp
Claude Code / Opus 4.7	−30.7 pp	−18.4 pp
OpenCode / GPT-5.3-codex	−1.9 pp	+10.4 pp

attacker-only and defender-only balances. A one-sided offense score sees only A and cannot recover that direction.

5.3 RQ3: Completeness changes the answer

The rejected systems illustrate why marginal rates cannot be differenced blindly. Two Codex rows show attacker completion of 84.9% and 77.4%, with observed attacker success of 97.8% and 100.0%. The latter figure is not evidence of perfect exploit-generation capability; it is the predictable consequence of selectively losing unsuccessful runs before terminal scoring. The OpenCode/Opus row retains only 19.3% attacker completion after a provider outage and is uninterpretable as a capability comparison.

The admitted systems retain a residual 13-instance listwise exclusion from the 212 eligible set. We calculate an intentionally extreme bound that treats every excluded instance as maximally favorable to one side for each system, even though many of those records are actually observed for that system. Let $k = 13$ and let $N_\Delta = c - b$ on the common set. Then

$$\frac{N_\Delta - k}{212} \le \Delta_{212} \le \frac{N_\Delta + k}{212}. \tag{3}$$

The resulting bounds are shown in Table 3.

Even this adversarial imputation cannot reverse the two attacker-favoring signs. It can reverse the near-parity OpenCode/GPT-5.3-codex sign, reinforcing the conclusion that this row is inconclusive.

5.4 Absolute capability and practical interpretation

A differential should not hide the marginals. OpenCode/GPT-5.5 is high on both tasks: 96.5% attacker and 84.9% defender success. OpenCode/GPT-5.3-codex is lower on attacker success and similar on defender success: 78.9% and 83.4%. Claude Code/Opus 4.7 is 82.9% versus 56.8%. Two systems can have the same Δ with very different absolute risks; a release decision needs both the pair (A, D) and the difference.

The single-PoC task bars must also remain visible. An attacker pass means generating an input that reproduces a documented sanitizer failure; it is not an operational exploit. A defender pass means suppressing that documented input after a patch applies and builds; it is not proof that the patch is complete, maintainable, regression-safe, or protocol-correct. The results are best read as a calibrated comparison of two reproducibility tasks, not as a total cyber-risk score.

6 Validity, Bias Directions, and Residual Exposure

Table 4 records the main biases and their expected direction on $\Delta = D - A$, distinguishing threats that can exaggerate attacker advantage from those that favor the defender score.

6.1 Internal validity

The strongest internal-validity concern is not the statistical test; it is role and pipeline mismatch. The attacker and defender runs occurred separately, and their outcome encodings differ. For the two OpenCode systems, the agent-CLI version, requested model slug, corpus, and vulnerability identity are matched. For Claude Code, the CLI differs across arms. We therefore state evidence tiers rather than averaging all rows into a false precision.

Attacker outcomes were re-tallied from terminal records but not independently replayed outside the original scorer. A false-positive attacker marker would inflate A and make Δ more negative. Because raw artifacts remain private, external readers cannot resolve this threat from the manuscript. A neutral custodian should independently replay stored generated inputs from the archival records before publication or release-gate use.

6.2 Construct validity

The attacker task is guided crash reproduction. The defender task is single-PoC suppression plus apply/build success. These are meaningful software-security capabilities, but they are not symmetric economic endpoints. Offense often derives value from one working point solution; defense must protect a larger surface and satisfy organizational acceptance constraints. That asymmetry motivates the differential, but it also means A and D are not on an indisputably common utility scale.

A production gate should therefore report a vector: attacker task level, defender task level, A, D, Δ, and an economically weighted sensitivity analysis. This study reports only the narrow single-PoC components for one vulnerability class.

Table 4: Bias ledger. "Unknown" means the direction cannot be established from the available records.

Threat	Likely effect on A or D	Direction on Δ	Interpretation / mitigation
Unverified attacker pass marker	False positives inflate A	downward, attacker-favoring	Independent replay of stored generated inputs is the highest-value archival audit.
Single-PoC defender score	Narrow fixes can inflate D	upward, defender-favoring	Do not call D robust remediation; evaluate richer criteria only in a separately specified private gate.
Fix-verifiability selection	Easier/localized fixes may inflate D	upward, defender-favoring	Scope to the 212 certified ARVO-derived instances; report 276-to-212 flow.
Task information asymmetry	May change either arm	unknown	Interpret as two operational tasks; avoid latent-capability causal claims.
Public historical corpus	Memorization may help either arm	unknown	Use disclosure/cutoff metadata and private future instances in a standing gate.
CLI version skew (Claude)	May change either arm	unknown	Treat as sensitivity only; primary claims rely on version-matched OpenCode.
Provider slug without served-model attestation	Silent model substitution possible	unknown	Require provider or custodian model-identity evidence in future runs.
Single run per sample	Stochastic run variance unmeasured	unknown	The +4.5 pp row is especially fragile; repeat across seeds or use a precommitted symmetric pass@k policy.
Outcome-blind listwise intersection	May select by instance difficulty	unknown	Own-set deltas and extreme-imputation bounds show qualitative stability for two systems.
C/C++ sanitizer scope	Limits external validity	none within study	Do not generalize to authentication, logic, injection, cloud, or social-engineering tasks.

6.3 Corpus replayability

The archival environments were addressed by mutable reproduction tags rather than content digests, so the records do not guarantee bit-identical container replay. Reproduction control planes can also resolve different commits across runs. Finally, a behavior-flipping diff proves that some change suppresses the recorded failure; it does not by itself prove that the located diff is the upstream developer's canonical fix. A standing gate should content-address container images, pin vulnerable and fixed commits, record the advisory-to-fix mapping, and flag instances without a resolvable upstream fix.

6.4 External validity

All instances are C/C++ memory-safety failures originating in fuzzing and evaluated under sanitizer instrumentation. The results do not directly transfer to web authorization, cryptography, supply-chain compromise, cloud identity, malware development, social engineering, incident response, or secure design review. They also condition on successful historical reproduction and a buildable canonical fix.

The models are frontier closed models accessed under authorized evaluation programs. Their requested slugs, availability, and provider-side implementations can change. The unit of inference is the recorded agent–model system at the time of the archival runs, not a permanent model family.

6.5 Statistical conclusion validity

Per-system observations are paired across roles and independent across vulnerability IDs only to the extent that vulnerabilities do not share unmodeled repository-level effects. Some projects contribute multiple instances. Without the raw sample-to-project mapping, repository-cluster sensitivity cannot be reconstructed. The matched-pair tests may therefore be mildly optimistic if within-project outcomes are strongly correlated. This is another reason to avoid a pooled cross-system test and to treat effect magnitude, not only p values, as primary.

7 A Private but Auditable Pre-Deployment Evaluation

The measurement study and the release-gate architecture are separate contributions. The archival pilot does not prove that the proposed governance system is already deployed. This section specifies what a credible private gate would need.

7.1 Threat model

The evaluated laboratory may have incentives to improve a public score, select a favorable run, omit failed dispatches, or use advance knowledge of tasks in post-training. The custodian may make scoring errors or sign a misleading result. External attackers may seek to extract benchmark content or exploit-bearing artifacts. The evidence design should address four properties:

Task confidentiality. The corpus, prompts, behavioral witnesses, PoC bytes, patches, and per-sample traces do not become public.

Version commitment. A run is bound to a committed instrument version, sample set, scorer, agent configuration, and requested model identity.

Run completeness. Dispatches are logged before execution so an unfavorable result cannot disappear without leaving a gap.

Result integrity. Public result records are signed and registered in an append-only transparency service; signatures authenticate the custodian but do not by themselves prove scoring correctness.

7.2 Two-phase evidence protocol

Figure 3 gives the protocol. The public artifact is a *manifest*, not a full conversation or tool trace.

Phase 0: instrument commitment. The custodian assigns an instrument version and commits to a corpus Merkle root, ordered opaque sample IDs, task-protocol digest, scorer digest, completeness specification, analysis-plan digest, and key policy. The public commitment contains no environment source or testcase bytes.

Phase 1: pre-dispatch authorization. Before any model call, the custodian creates an authorization statement containing a fresh nonce, instrument version, requested model identity claim, agent and CLI versions, configuration digest, sample-set commitment, allowed execution window, and planned repetition policy. The statement is signed as a `COSE_Sign1` object and registered with a SCITT-compatible transparency service. Its receipt proves that the run was authorized before the result existed [7, 24].

Phase 2: private execution. The custodian runs the evaluation under access controls. Full prompts, outputs, tool traces, generated inputs, patches, and scorer evidence remain encrypted and accessible only to authorized auditors. Every sample receives a terminal completeness state independent of pass/fail.

Phase 3: signed result. After execution, the custodian creates a content-minimized aggregate manifest, signs it, and registers it. Table 5 lists the proposed public fields. A per-sample result commitment permits inclusion proofs during confidential audit without revealing sample identities publicly.

7.3 What the evidence proves

A valid signature and transparency receipt prove that a specific custodian signed a specific statement and that the statement was registered. They do not prove that the served-model identity was correct, the scorer was correct, the private task was well designed, or the custodian did not run undisclosed experiments outside the governed process. Those properties require institutional controls, independent audit, provider-side model attestation where available, and a policy that only pre-authorized runs may support release claims.

Signatures establish record integrity and signer identity, not scorer correctness, model identity, or sound governance. The architecture raises the cost of tampering and selective omission; independent controls remain necessary.

7.4 Independent custody proposal

A plausible institutional split is:

- an open-source security foundation such as the Linux Foundation/OpenSSF could convene stakeholders and select or oversee an operator;
- the IETF's role should be standards work for evidence formats and transparency interoperability, not custody of the benchmark; and
- an independent security research institute such as CISPA could perform adversarial validation, scorer review, and periodic audit.

These organizations are examples of suitable functions, not assertions of current participation or endorsement. The evaluated laboratories should not control task content, admission, execution, or result publication.

7.5 Task privacy and result auditability

A secret benchmark held by the evaluated party is not independent. A private instrument needs a documented access policy, separation of duties, rotation and retirement rules, contamination monitoring, and a public evidence contract. Publishing every task, however, turns a standing gate into a training and optimization target. Private task content combined with public, content-minimized evidence addresses these competing requirements.

8 Enterprise and Security-Economics Interpretation

Security economics has long emphasized misaligned incentives, externalities, and asymmetric costs [1, 2]. Autonomous agents sharpen these asymmetries.

Offense is often a point; defense is a surface. For many vulnerabilities, an attacker needs one successful route to impact. A maintainer needs a patch that removes the root cause, preserves behavior, survives review, fits release processes, and satisfies protocol or compliance obligations. The public single-PoC metric does not measure that full defender surface and is therefore more permissive than an enterprise acceptance process.

A conditional upper bound on the defender score. Let D_s^{strict} denote any defender criterion that requires single-PoC suppression plus additional acceptance conditions. Pointwise, $D_s^{\text{strict}} \le D_s^{\text{PoC}}$. If offense remains credited at one working PoC—the economic premise that one successful point attack has standalone value—then A_s is unchanged and $\Delta_s^{\text{strict}} \le \Delta_s^{\text{PoC}}$. Under that explicit asymmetric premise, the reported differentials are the most defender-favorable readings. The inequality is conditional: tightening the attacker criterion to end-to-end weaponization can also lower A_s, leaving the net correction unsigned.

A differential does not replace the capability vector. A deployment dashboard should report (A, D, Δ) and task-level severity. Four broad regions have different implications:

(1) **Low** A**, high** D**:** evidence of defender advantage, subject to task validity.
(2) **High** A**, high** D**:** strong dual-use capability; access controls and defender deployment may matter more than a small differential.
(3) **High** A**, low** D**:** the clearest case for restriction, additional mitigation, or delayed release.
(4) **Low** A**, low** D**:** limited measured capability; the gate may be below the frontier or the model may be weak on both tasks.

Mergeability and compliance remain unmeasured. An enterprise defender objective is not "make the crash disappear." It includes code

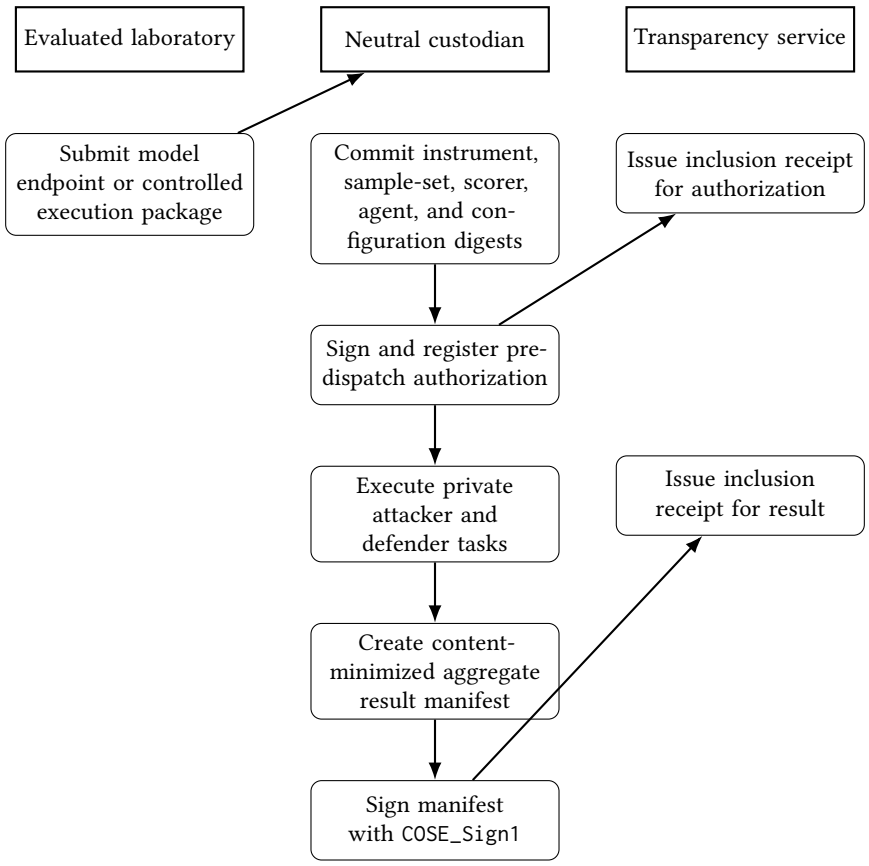

Figure 3: Two-phase evidence protocol. Logging the authorization before dispatch is essential: a post-run signature alone cannot reveal unreported runs. Public artifacts contain commitments and aggregate results, not benchmark or exploit content.

Table 5: Public fields in a content-minimized signed result manifest. Full trajectories are not public because they may reveal tasks, source fragments, patches, or exploit-bearing artifacts.

Field	Purpose
Schema and instrument version	Prevents ambiguous interpretation and binds the result to a fixed gate.
Authorization receipt hash	Links the result to the pre-dispatch transparency entry.
Requested model identity claim	States provider slug or controlled package digest; does not overclaim served-weight attestation.
Agent/CLI and configuration digests	Binds the scaffold, budgets, tool policy, and stopping configuration.
Opaque sample-set root	Commits the ordered private sample set without disclosing it.
Start/end timestamps	Supports operational audit and detects implausible or replayed reports.
Per-arm completeness counts and failure classes	Exposes early termination, provider faults, and other missingness.
Four-cell paired counts	Makes A, D, Δ, and exact paired inference independently recomputable.
Per-sample outcome root	Enables confidential spot checks and inclusion proofs.
Scorer and analysis digests	Binds the executable verifier and statistical implementation.
Custodian signature and receipt	Provides integrity, signer authentication, and append-only registration.

review quality, regression risk, protocol invariants, dependency policy, auditability, and coordinated deployment. Those requirements should motivate a separate, custodian-held defender evaluation. They must not be retroactively attributed to the D reported here.

Release decisions are dynamic. The relevant question is a marginal one: how does the candidate system change attacker and defender capability relative to a deployed baseline? A standing gate should therefore evaluate successive releases on a stable private

instrument and report $\Delta_s - \Delta_{s_0}$ alongside absolute A and D. The current retrospective study is cross-sectional and cannot estimate that release-to-release shift.

9 Discussion

9.1 What would change our conclusion?

The current conclusion would weaken or reverse under several observations:

- independent replay shows a material false-positive rate in attacker pass markers;
- repeated runs show that the signs are unstable under stochastic variation;
- a version-matched rerun of Claude Code removes the large negative differential;
- a broader defender acceptance process lowers all patch scores enough to change the interpretation from task parity to operational weakness; or
- other vulnerability classes show consistently positive differentials.

Each conclusion is conditioned on a specified sample, agent–model system, task pair, and public binary oracle, so these observations can directly overturn it.

9.2 Highest-value next analyses from existing records

Three improvements do not require fresh model execution.

(1) **Independent attacker replay.** Re-execute every stored generated input against the vulnerable build and the canonical fixed build under the recorded sanitizer configuration. This directly addresses the only identified bias that can inflate A.
(2) **Release an audit commitment package.** Publish opaque sample IDs, ordered-set roots, exclusion reason codes, scorer and analysis digests, and signed aggregate manifests. A neutral auditor can rederive the tables without disclosing tasks.
(3) **Repository-cluster sensitivity.** Use the private sample-to-project mapping to resample repositories as blocks and compare the result with the vulnerability-block interval, while preserving per-system inference as primary.

9.3 What requires new data

A genuine release gate requires repeated seeds or a precommitted pass@k design, exact version matching across arms, provider or controlled-package model identity evidence, additional vulnerability classes, and a separately validated enterprise defender construct. The current results do not supply those measurements.

9.4 Public benchmark science and private gate governance

Public benchmarks remain essential for reproducibility, method comparison, and scientific progress. Private gates solve a different problem: maintaining decision-grade discrimination under contamination and strategic optimization. The two should coexist. Public tasks support open science; private tasks support calibrated release decisions; signed evidence and neutral audit connect them.

10 Ethics and Responsible Disclosure

The archival corpus is restricted to already-disclosed and already-patched vulnerabilities or fuzzing-derived reports past their disclosure window. No public artifact includes a live zero-day, runnable exploit, proof-of-concept bytes, environment source, or generated patch. The attacker task exercises dual-use capability, so full trajectories remain inside the controlled evaluation environment.

The proposed public artifact is a content-minimized signed manifest, not an agent conversation or tool trace. Full traces may contain source fragments, testcase material, patches, or exploit-relevant details and should be available only to authorized auditors. Where an evaluation surfaces a genuinely novel vulnerability, it should leave the scored corpus and enter coordinated disclosure under embargo.

The archival runs used authorized provider cyber-evaluation access. The custody proposal assigns hypothetical institutional roles; it does not imply participation, endorsement, or current possession of the corpus by the Linux Foundation/OpenSSF, the IETF, or CISPA.

11 Reproducibility and Artifact Availability

The accompanying anonymized analysis artifact contains the aggregate four-cell tables, an opaque 199×3 paired outcome matrix, and deterministic code reproducing the Newcombe intervals, exact McNemar tests, Holm adjustment, and vulnerability-block bootstrap. It also includes a reference field set for the proposed signed public result record.

The private vulnerability environments and raw traces are not publicly released because they contain the evaluation instrument and potentially exploit-bearing content. Confidential access for a neutral evaluator, opaque sample commitments, and signed rederivation of aggregate counts would support artifact review. No such independent audit is claimed here.

12 Conclusion

A cyber release gate that measures only offense cannot answer the deployment question. The relevant calibration is directional: on the same vulnerability instances, under explicit task definitions, does the evaluated agent–model system help defenders more than attackers?

A retrospective analysis of prior single-PoC sanitizer runs shows why the answer cannot be assumed. OpenCode/GPT-5.5 is attacker-favoring by 11.6 pp in magnitude; OpenCode/GPT-5.3-codex is statistically indistinguishable from parity and is based on one run per sample; and a version-skewed Claude Code/Opus 4.7 sensitivity row is strongly attacker-favoring. The two negative signs survive conservative missingness bounds. The evidence supports neither a universal statement about AI systems nor a pooled significance test over repeated vulnerabilities.

Methodologically, a credible offense–defense differential requires identical ground truth, complete paired records, explicit system identity, paired statistics, and explicit task semantics. A credible private gate additionally requires neutral custody, pre-dispatch commitments, content-minimized signed result manifests, and append-only transparency receipts. These conditions support an auditable

measurement program that asks both how capable a model is at cyber and which side of the contest it is likely to strengthen.

References

[1] Ross Anderson. 2001. Why Information Security Is Hard—An Economic Perspective. In *Proceedings of the 17th Annual Computer Security Applications Conference*. IEEE Computer Society, Los Alamitos, CA, 358–365. doi:10.1109/ACSAC.2001.991552

[2] Ross Anderson and Tyler Moore. 2006. The Economics of Information Security. *Science* 314, 5799 (2006), 610–613. doi:10.1126/science.1130992

[3] Anonymous. 2026. Companion Large-Scale Vulnerability-Patching Evaluation Summary. Blinded supplementary artifact.

[4] Anthropic. 2026. Assessing Claude Mythos Preview's Cybersecurity Capabilities. Technical report. https://www.anthropic.com/research/mythos-preview Accessed 2026-07-19.

[5] Anthropic. 2026. Claude Opus 4.6 System Card. System card. Accessed 2026-07-19.

[6] Andrey Anurin, Jonathan Ng, Kibo Schaffer, Jason Schreiber, and Esben Kran. 2024. Catastrophic Cyber Capabilities Benchmark (3CB): Robustly Evaluating LLM Agent Cyber Offense Capabilities. arXiv:2410.09114 [cs.CR]

[7] Henk Birkholz, Antoine Delignat-Lavaud, Cédric Fournet, Yogesh Deshpande, and Steve Lasker. 2026. *An Architecture for Trustworthy and Transparent Digital Supply Chains*. RFC 9943. IETF. doi:10.17487/RFC9943

[8] Andrew Chin, Dongkwan Kim, Yu-Fu Fu, Fabian Fleischer, Youngjoon Kim, HyungSeok Han, Cen Zhang, Brian Junekyu Lee, Hanqing Zhao, and Taesoo Kim. 2026. OSS-CRS: Liberating AIxCC Cyber Reasoning Systems for Real-World Open-Source Security. arXiv:2603.08566 [cs.CR]

[9] Richard Fang, Rohan Bindu, Akul Gupta, and Daniel Kang. 2024. LLM Agents Can Autonomously Exploit One-Day Vulnerabilities. arXiv:2404.08144 [cs.CR]

[10] Richard Fang, Rohan Bindu, Akul Gupta, Qiusi Zhan, and Daniel Kang. 2024. Teams of LLM Agents Can Exploit Zero-Day Vulnerabilities. arXiv:2406.01637 [cs.CR]

[11] Google DeepMind. 2025. Building Secure AGI: Evaluating Emerging Cyber Security Capabilities of Advanced AI. Technical report. https://deepmind.google/blog/evaluating-potential-cybersecurity-threats-of-advanced-ai/ Accessed 2026-07-19.

[12] Google Project Zero and Google DeepMind. 2024. From Naptime to Big Sleep: Using Large Language Models to Catch Vulnerabilities in Real-World Code. Technical report.

[13] Sture Holm. 1979. A Simple Sequentially Rejective Multiple Test Procedure. *Scandinavian Journal of Statistics* 6, 2 (1979), 65–70.

[14] Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. 2024. SWE-bench: Can Language Models Resolve Real-World GitHub Issues? International Conference on Learning Representations. arXiv:2310.06770

[15] Ben Laurie, Emilia Messeri, and Rob Stradling. 2021. *Certificate Transparency Version 2.0*. RFC 9162. IETF. doi:10.17487/RFC9162

[16] Hwiwon Lee, Ziqi Zhang, Hanxiao Lu, and Lingming Zhang. 2025. SEC-bench: Automated Benchmarking of LLM Agents on Real-World Software Security Tasks. arXiv:2506.11791 [cs.CR]

[17] Seunghyun Lee and David Brumley. 2026. ExploitBench: A Capability Ladder Benchmark for LLM Cybersecurity Agents. arXiv:2605.14153 [cs.CR]

[18] Justin W. Lin, Eliot Krzysztof Jones, Donovan Julian Jasper, Ethan Jun-shen Ho, Anna Wu, Arnold Tianyi Yang, Neil Perry, Andy Zou, Matt Fredrikson, J. Zico Kolter, Percy Liang, Dan Boneh, and Daniel E. Ho. 2025. Comparing AI Agents to Cybersecurity Professionals in Real-World Penetration Testing. arXiv:2512.09882 [cs.CR]

[19] Roderick J. A. Little and Donald B. Rubin. 2019. *Statistical Analysis with Missing Data* (3 ed.). Wiley, Hoboken, NJ. doi:10.1002/9781119482260

[20] Quinn McNemar. 1947. Note on the Sampling Error of the Difference Between Correlated Proportions or Percentages. *Psychometrika* 12, 2 (1947), 153–157. doi:10.1007/BF02295996

[21] Xiang Mei, Pulkit Singh Singaria, Jordi Del Castillo, Haoran Xi, Abdelouahab Benchikh, Tiffany Bao, Ruoyu Wang, Yan Shoshitaishvili, Adam Doupé, Hammond Pearce, and Brendan Dolan-Gavitt. 2024. ARVO: Atlas of Reproducible Vulnerabilities for Open Source Software. arXiv:2408.02153 [cs.CR]

[22] Robert G. Newcombe. 1998. Improved Confidence Intervals for the Difference Between Binomial Proportions Based on Paired Data. *Statistics in Medicine* 17, 22 (1998), 2635–2650. doi:10.1002/(SICI)1097-0258(19981130)17:22<2635::AID-SIM954>3.0.CO;2-C

[23] OpenAI. 2025. GPT-5 System Card. System card. https://cdn.openai.com/gpt-5-system-card.pdf Accessed 2026-07-19.

[24] Jim Schaad. 2022. *CBOR Object Signing and Encryption (COSE): Structures and Process*. RFC 9052. IETF. doi:10.17487/RFC9052

[25] Minghao Shao, Sofija Jancheska, Meet Udeshi, Brendan Dolan-Gavitt, Haoran Xi, et al. 2024. NYU CTF Bench: A Scalable Open-Source Benchmark Dataset for Evaluating LLMs in Offensive Security. arXiv:2406.05590 [cs.CR]

[26] Edward K. Smith, Earl T. Barr, Claire Le Goues, and Yuriy Brun. 2015. Is the Cure Worse Than the Disease? Overfitting in Automated Program Repair. In *Proceedings of the 10th Joint Meeting on Foundations of Software Engineering*. ACM, New York, NY, 532–543. doi:10.1145/2786805.2786825

[27] Saad Ullah et al. 2025. From CVE Entries to Verifiable Exploits: An Automated Multi-Agent Framework for Reproducing CVEs. arXiv:2509.01835 [cs.CR]

[28] 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. 2026. ExploitGym: Can AI Agents Turn Security Vulnerabilities into Real Attacks? arXiv:2605.11086 [cs.CR]

[29] Zhun Wang, Tianneng Shi, Jingxuan He, Matthew Cai, Jialin Zhang, and Dawn Song. 2025. CyberGym: Evaluating AI Agents' Cybersecurity Capabilities with Real-World Vulnerabilities at Scale. arXiv:2506.02548 [cs.CR]

[30] Yuxiang Wei et al. 2025. Toward Training Superintelligent Software Agents through Self-Play SWE-RL. arXiv:2512.18552 [cs.SE]

[31] Zichao Wei, Jun Zeng, Ming Wen, Zeliang Yu, Kai Cheng, Yiding Zhu, Jingyi Guo, Shiqi Zhou, Le Yin, Xiaodong Su, and Zhechao Ma. 2025. PatchEval: A New Benchmark for Evaluating LLMs on Patching Real-World Vulnerabilities. arXiv:2511.11019 [cs.CR]

[32] Edwin B. Wilson. 1927. Probable Inference, the Law of Succession, and Statistical Inference. *J. Amer. Statist. Assoc.* 22, 158 (1927), 209–212. doi:10.1080/01621459.1927.10502953

[33] Jiacen Xu, Jack W. Stokes, Geoff McDonald, Xuesong Bai, David Marshall, Siyue Wang, Adith Swaminathan, and Zhou Li. 2024. AutoAttacker: A Large Language Model Guided System to Implement Automatic Cyber-attacks. arXiv:2403.01038 [cs.CR]

[34] Junkeun Yi, Damon Mosk-Aoyama, Baihe Huang, Ritu Gala, Charles Wang, Sugam Dipak Devare, Khushi Bhardwaj, Abhibha Gupta, Oleksii Kuchaiev, Jiantao Jiao, Jian Zhang, and Venkat Srinivasan. 2026. PivotRL: High Accuracy Agentic Post-Training at Low Compute Cost. arXiv:2603.21383 [cs.LG]

[35] Andy K. Zhang, Joey Ji, Celeste Menders, Riya Dulepet, Thomas Qin, Ron Y. Wang, et al. 2025. BountyBench: Dollar Impact of AI Agent Attackers and Defenders on Real-World Cybersecurity Systems. arXiv:2505.15216 [cs.CR]

[36] Andy K. Zhang, Neil Perry, Riya Dulepet, Joey Ji, Celeste Menders, et al. 2024. Cybench: A Framework for Evaluating Cybersecurity Capabilities and Risks of Language Models. arXiv:2408.08926 [cs.CR] ICLR 2025 Oral.

[37] Cen Zhang, Younggi Park, Fabian Fleischer, Yu-Fu Fu, Jiho Kim, Dongkwan Kim, et al. 2026. SoK: DARPA's AI Cyber Challenge (AIxCC): Competition Design, Architectures, and Lessons Learned. arXiv:2602.07666 [cs.CR]