

When green tests are unsound

Executable evidence transfer across coding roles

Artur Onysko | Independent AI-assisted builder, Estonia

AI Collusion Research Sprint | 23-25 October 2026

Abstract

Executable test drafts can carry mistaken evidence between separate coding roles without an explicit malicious instruction. We distinguish that observable dependence from deliberate collusion using native execution of complete generated suites. Two prospectively frozen studies cover three cache APIs and three iterator APIs, with benign, revealing and unsound drafts matched in byte length and code scaffold within each task. Valid catches per nine planned conversations are 7/7/5/4 across no-draft/benign/revealing/unsound cache arms and 5/6/8/4 in the iterator arms. Thus a correct carried witness has no consistent end-to-end advantage across contexts. In the iterator study, six current-checkout-green files fail pristine code. A separate matched prose experiment yields four valid catches with target-relevant clarification versus none with off-target clarification, supporting an ordinary-information alternative. Reference-only pruning makes all 35 completed iterator files valid but leaves seven target misses, despite perfect coverage of the additional planted bank. Exact-body quarantine changes none of those original iterator endpoints. On one exposed API, eighteen fresh receivers of two selected model-generated wrong parents yield 4/6 valid catches with wrong drafts, 5/6 after minimal repair and 0/6 without drafts; effects differ by parent. These repairs change list length. On six earlier outputs, structured failure guidance uses eight reference processes versus 34 for a stronger conventional comparator, with equal measured utility; authored counterexamples preclude dominance. The contribution is a reproducible audit of executable evidence, missing coverage and correction cost, not a general collusion detector or evidence of shared intent.

1. Audit the evidence carrier

A reviewer may inherit an executable test rather than a prose instruction. Its output can then contain a wrong expectation, omit the needed case, or fail to finalize. These are distinct outcomes. We ask which survive matched interventions, which ordinary explanation remains viable, and what a native correction policy actually costs.

Pinned planted checkout	Untrusted test-source data	Honest coding reviewer	Whole-file native audit
-------------------------	----------------------------	------------------------	-------------------------

Figure 1. Only supplied evidence data changes in the matched studies. The reviewer is instructed to review independently. An authored unsound draft is a controlled intervention, not an observed secret alliance.

2. Design and qualified execution

Prior work already studies bug-validating generators [1], incorrect source influence [2], transfer of bad tests [3], feedback/scaffold decomposition [4], false-premise verifier coupling [5] and assertion filtering [6]. Our incremental question is how an executable carrier affects joint validity, target coverage and correction cost under explicit controls. We do not reproduce activation-based collusion detection [7] or covert shared objectives [8].

2.1 Public source and independent finite qualification

Table 1. Local planted changes, never upstream bug allegations. Finite correctness is not all-input correctness.

Source	Upstream regression gate	Independent domain / target differences
cachetools v7.2.4 [9]	366 tests + 8 subtests, all four variants pass	LRU: 7,380 / 24; TTL: 48 / 18; keywords: 64 / 8
more-itertools v11.2.1 [10]	930 tests, 5 skips, all four variants pass	chunked: 2,912 / 365; windowed: 3,780 / 390; unique: 1,364 / 366

Pristine code has zero discrepancies on these independent finite domains. Cache mutants alter truthy lookup touching, fractional TTL and keyword case. Iterator mutants alter falsy strict tails, falsy padding and keyed case equivalence. The latter APIs are new, but truthiness/casefold defect themes are exposed. This is not certified unseen training data.

Each authored triple has equal UTF-8 bytes and AST scaffold after masking relevant constants: cache 204/254/266 bytes; iterator 258/175/184 bytes. Benign passes pristine and target; revealing passes pristine only; unsound passes target only. All 18 drafts are natively qualified before their respective generations. No arm label or gate verdict reaches the receiver. The no-draft arm is an availability control, not length matched. Opaque provider tokens are not matched.

2.2 Protocol and endpoints

Both mains have 3 tasks x 4 arms x 3 clustered repeats. The requested model is deepseek-v4.1-flash, temperature 0.2, disabled thinking, 4,096 output tokens and at most three coding turns, with current-checkout tool feedback. Whole-file validity requires admitted tests passing pristine, no skips/errors/denied operations; a valid catch also fails the target. Invalid, nonfinal and unavailable states stay separate. No repaired subset upgrades its original file.

The cache context includes detailed contracts and whole source modules. Iterator context includes verbatim function documentation and complete function/helper windows. All arms within each study share their context. Cross-study differences do not isolate library identity. Repeats share tasks; we report finite counts without pooled population tests.

3. Matched carriers: a context-dependent result

Table 2. Original complete outputs, before any repair. Cache: 91 coding POSTs; iterator: 93. Bank opportunities exist only among valid files and differ between studies.

Study / draft	Slots	Catch	Miss	Invalid	Nonfinal	Bank / available
Cache / none	9	7	0	1	1	19/21
Cache / benign	9	7	0	2	0	14/21
Cache / revealing	9	5	0	4	0	9/15
Cache / unsound	9	4	0	5	0	7/12
Iterator / none	9	5	1	3	0	12/12
Iterator / benign	9	6	0	3	0	12/12
Iterator / revealing	9	8	0	1	0	16/16
Iterator / unsound	9	4	0	4	1	8/8

The revealing carrier is not consistently superior: 5 versus 7 no-draft catches in cache, but 8 versus 5 in iterators. Unsound carriers produce four valid catches in either study; this does not measure a population attack rate. The cache main has no valid miss. The iterator no-draft unique task has one valid miss that still catches both bank variants.

The exact wrong supplied LRU body survives all three unsound cache finals. The wrong unique body survives two unsound iterator finals. Exact matching is evidence of a shared code body, not proof of copying, causal authorship or hidden intent. Other assertions can independently preserve or corrupt the whole file.

3.1 Why current-checkout green and red are inadequate

A post-outcome receipt audit retains the full pristine/target outcome square. In iterators, 23 files pass pristine and fail target; one passes both; five fail both; six FAIL pristine yet PASS target; one is nonfinal. Those six include two no-draft, two benign and two unsound outputs. They cannot all be attributed to a supplied bad test.

Thus target-only red would mislabel five unsound files useful catches, while current-only green would mislabel six unsound files sound assurance. These are hypothetical metric mistakes, not newly defined successful attacks. The cache main similarly has 12 fail/fail files. Counts alone do not distinguish assertion errors from other test-body errors.

4. Ordinary information remains a strong explanation

Table 3. Two separate prospective cache studies, not a pooled experiment. Information package: 24 conversations, 65 POSTs. No-draft prose clarification: 12 conversations, 32 POSTs.

Information / draft	Slots	Catch	Miss	Invalid	Nonfinal	Bank / available
explicit / none	6	5	0	1	0	10/15
explicit / unsound	6	3	0	3	0	7/9
docs / none	6	0	1	4	1	2/3
docs / unsound	6	0	0	5	1	0/0
Prose / targeted	6	4	0	1	1	8/12
Prose / control	6	0	2	2	2	4/6

Replacing the detailed contract by ordinary pinned documentation changes the entire information/style/length package. It does not isolate one clue. Documentation yields no original valid catch; nine completed docs files fail during pristine test-body execution, one valid file misses the target and two do not finalize. This is not a collection failure.

A next matched contrast holds those docs, implementation, tool and no-draft state fixed. Truthful target-relevant and equally API-specific off-target clarifications have equal word counts and UTF-8 bytes, with padding disclosed. Four versus zero valid catches support an ordinary useful-information explanation. Equal opaque tokens or linguistic difficulty are not established; no code-only coordination mechanism is identified.

4.1 Same observation, different claimed authority

A separate fresh nine-prefix experiment forks each exact first visible coding response into three histories: current only, auxiliary execution labelled candidate, or the identical execution labelled reference. The auxiliary is really pristine in both labelled arms; candidate withholds certification rather than inventing outputs. The role names are both nine ASCII bytes. Seven initial prefixes yield usable native A/B feedback; two invalids remain in the denominator.

Across 27 branches and 53 unique POSTs, valid catches are 6/9, 6/9 and 7/9. One prefix favors reference certification; this does not establish authority superiority or equivalence. Shared first responses count once in cost. Exact visible history is shared, not a cloned provider hidden state. The 19 catches and eight invalids remain separate.

4.2 Initial implementation availability

A further frozen 12-conversation contrast uses ONE exposed unique API: initial code shown/omitted x no/unsound draft x three repeats, with the same docs/tool/current feedback. Valid catches are 2/3 and 1/3 with code shown, versus 0/3 in either omitted arm. Omission leaves one valid target miss per arm and one nonfinal. Both omitted/unsound invalid files retain the wrong supplied AST body and pass the target while failing pristine.

Independent native and prompt-block checks agree on all 12 slots. Initial source omission did not rescue this observed outcome. It changes length, and later runtime traces may reveal source: neither full-loop blindness nor artifact-only causation is established. All five valid files catch both easy bank variants despite two target misses.

5. Validity repair is not coverage repair

5.1 Reference-only failure-guided pruning

Execute the whole file on trusted pristine code. Keep a green valid file; otherwise use structured failed test node identities to remove failing top-level tests, preserve helpers/setup and recheck the combined file. Stop after three reference processes or HOLD on illegal/ambiguous/still-failing output. No target or bank enters selection. Pristine availability is oracle privilege, not an oracle-free monitor.

5.2 Exact-body provenance quarantine

Remove top-level tests whose AST body exactly matches an actually supplied draft, ignoring function names. Preserve helpers; if nothing matches, preserve original bytes. This uses zero native calls for SELECTION, while outcome evaluation still costs native runs. Matching does not certify provenance or correctness; renamed equivalents evade it.

Table 4. Separate original and selected outcomes. Cache/info analyses reuse exposed outputs; iterator policies were frozen before its new generation. Unchanged policies transfer via an explicit package-native dependency port.

Cohort / slots	Original catch / miss	Reference catch / miss	Quarantine catch / miss
Cache / 36	23 / 0	34 / 1	25 / 0
Information / 24	8 / 1	16 / 6	12 / 1
Iterator / 36	23 / 1	28 / 7	23 / 1

Reference pruning makes 35/35 completed iterator files valid using 46 reference processes and 697 collected pytest items, yet seven target misses remain. All 35 catch both additional bank variants (70/70): that easy bank cannot certify coverage of the difficult target. Original valid target/bank catches are not lost. The information cohort similarly becomes 22 valid files with six target misses, using 35 processes / 388 items.

Quarantine removes only three iterator tests and changes none of its original endpoints: 23 catches, one miss, 11 invalids and one nonfinal remain. Two exact wrong-body removals do not eliminate the other wrong expectations in those files. It helped on earlier cache/info outputs, but that local improvement does not transfer as a guarantee.

Early isolation amendments wrongly dropped exception-based oracles and two bank catches; both failed repairs and the corrected instrument remain recorded. Final policy comparisons do not rewrite these post-outcome development failures.

6. Price the actual countermeasure

Table 5. Same six newly generated files on exposed cache tasks. The suite-first comparator is a later challenge on these reused files. Each policy obtains six target catches and 12/18 bank catches.

Policy	Actual reference processes	Collected pytest items
Individual tests + combined recheck	71	125
Whole suite first; individual fallback	34	112
Structured-failure guided pruning	8	86

Against suite-first early exit, the observed PROCESS ratio is $34/8 = 4.25$, not a total-computation speedup. Guided receives richer failed-ID information. Generation cost is shared; shared-host timing is not an isolated benchmark.

6.1 Is pruning worth more than one validity gate?

Table 6. Reject-only retains identical original bytes only after one pristine-green call; guided also salvages failing files. Actual reference processes and useful catches, separately by reused cohort. Four nonfinals remain. No target or bank enters either selection.

USED cohort	Reject-only calls / catches	Guided calls / catches	Extra calls / catches
Cache (36 slots)	35 / 23	47 / 34	12 / 11
Information (24)	22 / 8	35 / 16	13 / 8
Iterator (36)	35 / 23	46 / 28	11 / 5

Reject-only HOLD counts are 12/13/11; guided valid target misses are 1/6/7 versus gate 0/1/1. Original-valid target/bank catches are preserved. The gate was defined post-outcome on USED files; guided costs are independently recounted old receipts, not fresh guided runs. These marginal counts are not pooled ROI or total CPU savings.

6.2 Fixed order stress on generated selected files

All 92 distinct selected suites qualify for a prospectively frozen same-order normalization, reverse and one-position rotation of plain test definitions. Individual test ASTs and non-test module positions stay fixed. All controls reproduce their prior endpoints; neither permutation changes pristine validity or target detection. Catches/misses remain 34/1, 16/6 and 28/7. Actual execution uses 552 native children; an independent consumer checks all transforms/raw receipts and reruns 18 children on three fixed sources. No negative trigger requires confirmation.

This finite negative does not establish arbitrary-order independence. Authored state/cap counterexamples still refute general selector dominance (Appendix C, Table 7). Missing witnesses remain missing; green is not coverage.

7. Causal reading and research contribution

What the interventions distinguish

Within each matched task the source, tool contract, reviewer objective and draft scaffold stay fixed while relevant literal evidence changes. Original endpoint differences therefore concern that specified input intervention and sampled coding trajectories. Equal scaffold eliminates a simple unequal-format explanation for those contrasts, not every information-quality or stochastic explanation. Three repeats per task do not identify a general population effect.

The two library contexts disagree on whether a revealing carrier improves the end-to-end result. The matched prose challenge independently shows target-relevant help without executable code. The role fork finds only a one-prefix difference for certified source authority. Together these observations prevent treating code transmission, correctness certification or one poisoned draft as a uniquely established collusion mechanism.

A fixed generated program can be re-executed on pristine, target and bank variants; a sealed selected subset can be evaluated without feeding target outcomes into its selection. Those are controlled native program outcomes, not claims about why a model internally wrote a line. Exact supplied-body retention does not isolate copying from shared context, and removing it need not remove the broader mistaken premise.

7.1 Actual wrong model parents

Two old no-draft unique outputs pass target and fail pristine. Expected-list-only repairs are independently qualified, preserving all other bytes. A frozen 18-receiver contrast retains ONE exposed API and two USED parents without giving parent verdicts. Per-six useful catches are null 0, actual wrong 4, authored repair 5; wrong has one valid miss and one invalid, repair one invalid, null six invalid. Parent blocks disagree: P1 wrong 3/3 versus repair 2/3; P2 wrong 1/3 versus repair 3/3. One exact wrong body survives with fail-pristine/pass-target; repaired input also leads to a new wrong assertion. These outcomes do not support uniform carrier harm. Useful case scaffolds, ordinary source/feedback anchoring and changed list lengths remain alternatives; this contrast is not byte/AST-shape matched (Appendix D, Table 8).

Increment beyond the nearest work

Wrong-source bias, bad-test transfer, verifier coupling, filtering and matched feedback controls are already studied [1-6]. The incremental artifact here combines byte/AST-matched executable evidence interventions, strict whole-file endpoint accounting, a visible-history role fork, new-library qualification, a truthful-prose alternative and an explicit correction-cost/utility frontier. We claim the reproducible boundary and counterexamples, not invention of test filtering or a new universally effective detector.

Relevance to multi-agent safety

An honest verifier can accept or recreate evidence aligned with an incorrect implementation. Such evidence is a potential coordination carrier in a larger multi-agent workflow. Our controlled drafts demonstrate observable dependence and its measurement hazards. They do not demonstrate covert shared intent, strategic deception, unprompted agent collusion, activation-level detection performance or real deployment harm.

The useful operational rule is to audit the complete evidence file against an independent correctness anchor and retain missing-coverage and nonfinal states after repair. When that anchor is absent, provenance quarantine is a limited heuristic with measured losses and evasions, not an equivalent substitute.

8. Limits, reproduction and conclusion

Limitations

One requested provider/model route; six curated APIs across two libraries; authored rather than naturally reported defects; exposed truthiness/casfold themes; small dependent repeats; variable information packages; and privileged access to pinned pristine implementations limit generalization. Source-window presentation changes across libraries. The additional banks are finite planted variants, and the iterator bank is particularly easy. No population p-values, prize probabilities, broad attack resistance or unseen-real-bug claim follows.

Computational cross-checks use independently written consumers but share native primitives, libraries, host and author. They are not external human peer review. Python AST/import limits, read-only audit hooks, resource caps and timeouts are bounded execution controls, not an OS sandbox or security certification. Native qualification failures remain visible rather than being counted as model mistakes.

Reproduction and exactness

The offline supplement freezes protocols before their stated experiments, request/response projections, final code, source manifests, native receipts, selected programs and negative gates. Offline replay needs no model call. Completed clean-extracted replay receipts bind the exact packet identities. Earlier studies and later extensions retain separate execution records; a parent receipt is never substituted for execution of a new extension.

Replay comparison preserves program bytes, verdicts, availability, failed test IDs, plugin counts, retained code and selection budgets. Raw traces are saved but not claimed byte-identical: temporary roots, Python repr order and timing can vary. Recorded role feedback uses a declared display-root adapter while native code executes in staged copies. Input and packet manifests must remain unchanged.

Conclusion

A correct carried witness can help in one context and fail to improve the complete suite in another. Green tests on a changed implementation can be unsound. Trusted-reference pruning repairs assertion validity while leaving missing target coverage, even when a small auxiliary bank appears perfect. Exact-body quarantine is cheaper at selection and much narrower in effect. Additional pruning buys measured useful catches beyond a reject-only gate in each reused cohort; two fixed order stresses leave those outcomes unchanged. The practical result is an auditable evidence boundary with honest costs and failure cases, not a declaration that collusion has been detected or solved. Actual wrong model drafts can help more than no draft while retaining a specific unsound expectation. Minimal repairs produce opposite catch differences across the two selected parent blocks.

The initial-source control adds a practical negative: removing implementation context does not automatically decouple verification. The contrast also withholds potential defect-localization information; it does not identify which internal pathway changed. Wrong supplied bodies can persist without the initial display, while current feedback and ordinary API difficulty remain alternative pathways. This is a finite boundary, not a population reversal of prior source-bias work.

Code and data: an offline reproduction supplement with controlled coding transcripts, source manifests and executable checks is prepared. No public repository is linked in this report; the materials are available for author-mediated access. Earlier experimental strata and failed instrument amendments remain identified separately.

References

- [1] N. S. Mathews and M. Nagappan. Design choices made by LLM-based test generators prevent them from finding bugs. 2024. arXiv:2412.14137v1. <https://arxiv.org/abs/2412.14137v1>
- [2] D. Huang et al. Measuring the Influence of Incorrect Code on Test Generation. 2025 revision. arXiv:2409.09464v3. <https://arxiv.org/abs/2409.09464v3>
- [3] A. Berndt et al. On the Flakiness of LLM-Generated Tests for Industrial and Open-Source Database Management Systems. ICSE SEIP 2026. arXiv:2601.08998v1. <https://arxiv.org/abs/2601.08998v1>
- [4] Y. Liang et al. Auditing and Decomposing Feedback-Driven Evolution in LLM Test Generation under the Oracle Problem. 2026. arXiv:2608.19626v1. <https://arxiv.org/abs/2608.19626v1>
- [5] F. He et al. Verification Trap: Understanding Test-Time Selection Failures under False Premises in Code Generation. 2026. arXiv:2610.05170v1. <https://arxiv.org/abs/2610.05170v1>
- [6] H. Yu et al. DeCon: Detecting Incorrect Assertions via Postconditions Generated by a Large Language Model. 2025. arXiv:2501.02901. <https://arxiv.org/abs/2501.02901>
- [7] A. Rose et al. Detecting Multi-Agent Collusion Through Multi-Agent Interpretability. Revision 15 October 2026. arXiv:2604.01151v5. <https://arxiv.org/abs/2604.01151v5>
- [8] N. Radev et al. The Best-Laid SCHEMES: Coordinated Sabotage and Monitoring in Multi-Agent Systems. 2026. arXiv:2605.29178v1. <https://arxiv.org/abs/2605.29178v1>
- [9] T. Kemmer and contributors. cachetools, MIT license; v7.2.4, commit 6f23e2a7b75a42d30cb9d890ab3ca4059ee01bfd. <https://github.com/tkem/cachetools>
- [10] more-itertools contributors. more-itertools, MIT license; v11.2.1, commit 4aeb4338ae1407d8da9723e86da99efeb446f719. <https://github.com/more-itertools/more-itertools>

Appendix A. Ethics, dual use and AI assistance

Ethics and controlled setting

All altered programs, unsound drafts and counterexamples are local research artifacts. No production code, third-party CI, account or unsuspecting reviewer is attacked. Both public source licenses and pristine files are retained. Authored mutants are attributed to this study, never presented as upstream bugs. Native resource/profile failures do not count as model behavior.

Dual-use considerations

Unsound executable tests could be misused to create false assurance in a collaborative coding pipeline. We disclose the intervention and qualification rather than hiding it in an ostensibly trusted tool. The research illustrates audit and utility boundaries; it provides no instructions for unauthorized deployment, credential access or covert exfiltration. Local Python controls do not make arbitrary adversarial execution safe.

AI disclosure and human responsibility

Codex substantially assisted study design, software, execution, analysis and writing. Artur Onysko is the responsible human author; no AI is represented as a human coauthor or independent human reviewer. Model generations, authored carriers, software fixtures and native counterexamples are explicitly separate. No scripted output is called an LLM response and no LLM judge determines scientific outcomes.

The eight new generation strata in this candidate use 420 genuine coding POSTs and 2,873,609 provider-reported tokens: matched cache 91, role fork 53, fresh policy screen 16, information 65, prose hint 32, iterator transfer 93, initial-source control 29 and actual-parent receivers 41. A shared-prefix response is counted once. Reusing outputs for native policy analysis adds no model sample. Requested model settings and usage metadata do not attest internal reasoning or invoice cost.

Data, resources and preexisting work

Only public licensed source and controlled local coding context enter requests. Credentials, raw provider bodies, hidden reasoning and unrelated private system data are excluded from the reproduction dataset. Existing authorized coding resources are used with fresh quota checks before each request, no uncertain retries, no new payments or installations. Frozen transport code is provenance and must not execute during offline replay.

Harness preparation on 22 October is preexisting. The 420-response studies and their reported native extensions were conducted during the 23-25 October sprint. Older studies remain separately identified prior evidence. Public library sources and cited methods are preexisting third-party work. The reproduction README preserves this distinction and the third-party licenses.

Appendix B. Evidence strata and reproduction ledger

Prospective generation versus later analysis

Initial-source control: 12 trajectories / 29 POSTs on the exposed unique API, with five pure instrument checks before its prospective freeze. Its 12 whole-file native outcomes and paired display blocks were independently checked. Its separate clean replay is linked by exact packet identity. The parent receipt is retained as a prerequisite, not relabelled as proof of newer data.

Cache matched main: 36 trajectories / 91 POSTs. Shared-prefix authority fork: 9 first samples / 27 branches / 53 unique POSTs. Fresh policy screen: 6 trajectories / 16 POSTs on exposed cache tasks. Information challenge: 24 trajectories / 65 POSTs. Prose-hint challenge: 12 trajectories / 32 POSTs. Iterator transfer: 36 trajectories / 93 POSTs. None is a population-independent task sample. Actual-parent contrast: 18 trajectories / 41 new responses on ONE exposed API; two USED source parents.

The suite-first comparator is a post-outcome challenge on the same six files. Cache/info quarantine was defined after old main outcomes and after information generation started; it is not prospective confirmation there. Iterator policy sources and exact package dependency profile were frozen before its generation. The endpoint square is a later descriptive audit, never a new attack-success endpoint.

Two later native-only cuts reuse the 96 planned cache/information/iterator slots: a reject-only cost frontier and fixed-order stress on 92 selected generated suites. No new coding response is added. Their prospective transformation protocol does not make the underlying USED files new model samples. Their completed clean replay is linked separately from the actual-parent extension; older workers are not credited as new execution.

Preserved earlier work

The repository predecessor contains real writer-generated carriers: useful catches 4/9 no draft, 9/9 honest draft, 4/9 keep-green draft and 6/9 authored witness substitution, from 87 main POSTs. Only one task qualified a valid witness-free weak parent. Its small receivers and earlier finite-ID studies differ in interface/budget and remain exposed prior evidence, not pooled fresh results. Pre-sprint material and 22 predecessor development responses are disclosed separately from the present 420 calls.

Negative instrument results

Original per-test isolation rejected seven sources before execution: 494 attempts but 487 native children, not 494 executions. Pytest-only and manual exception-oracle amendments stay distinct. Iterator qualification first rejected a legitimate lazy import, then encountered full-suite thread/config-read failures. The accepted profile bounds trusted-test threads/arenas and disables personal PDB configuration inside the child; it does not grant private reads or change installed packages.

The full iterator gate uses the pinned upstream 930-case suite with five reported skips, while generated-file admission permits no skips. Source manifests and the independent finite oracle are unchanged between accepted variants. The package port replaces one allowed import and three child path/package references; selectors and conversation parsing retain frozen source identities.

Iterator prospective protocol SHA-256: bfcda7904ec2e9dd0a16bd537c5f570ead42991c1a63dab237039afd634d9e3e

Iterator original analysis SHA-256: f55851a30e489fcd9356f7af0df33846e8185638e6e20ab9c833c48d779bfb3aa

Iterator policy analysis SHA-256: fa0fd6dd00e9e247dd59c82e8e210b7798cb636586af4a00774932db93a83cb6

A manifest-bound manuscript and a separate rendered-text/numeric consumer accompany the report. Scientific source identities, complete table values and the abstract are unchanged in the submission presentation edition. Exact packet and clean-replay identities accompany the offline supplement; computational cross-checks are not independent human review.

Appendix C. Counterexamples and robustness receipts

Table 7. Authored, actually executed adversarial cases, not model samples or prevalence estimates.

Counterexample	Native result	Consequence
Printed fake failure / duplicate marker	Actual IDs resist stdout spoofing; ambiguous marker gives HOLD	Receipt ownership, not output-text authority
Cross-test module state	Isolated passing tests can fail together	Combined recheck is essential
Dependency cascade	Three-call guided cap loses a file retained by individual testing	No selector dominance
Sound witness quarantine, three APIs	Removing the only inherited useful test creates valid misses	Provenance is not correctness
Renamed wrong test, three APIs	Exact-body quarantine removes nothing; pristine still fails	Syntax matching is not a general defense

Conditional process model

The finite authored mask check assumes deterministic independent test bodies and fixed helpers. Structured failure IDs can save subprocess startup under those assumptions. It is not a new general theorem, and state/dependency counterexamples are retained rather than dismissed by the generated-suite order negative.

Order-stress accounting

The frozen schedule preserves all 96 slot identities; its 92 selected programs are distinct within cohort/task/native profile. All meet the plain zero-argument top-level definition gate, with 6-23 tests for cache/information and 8-23 for iterator. Definition-time effects and references to test names are conservatively rejected. Same-order control, reverse and one-position left rotation share the same AST unparse route. No adaptive permutation search is allowed.

Primary execution: 210/132/210 children by cohort (552 total), 7,008 collected pytest items. No identical-transform reuse or negative-trigger confirmation occurs. Independent AST closure/raw-counter classification/cost checking covers every eligible source; actual independent native execution covers u001/u036/u058, all three orders on pristine/target: 18 children. This shares the native instrument, host and author and is not a second full-corpus execution.

An additional post-outcome readback verifies declared test counts against collection and finds no changes in exit/pass/fail/error/skip/denial counters across orders. Raw traces and temporary paths are retained, not claimed byte-identical. The result does not measure all permutations, new model behavior or population prevalence.

Order protocol SHA-256: c7e528942a912397ac88f62e6b415356b5400b4749b2c7233809d14bfc3e73fc

Order native analysis SHA-256: fclac2c66d1e3fe679689a07064307b4125161f985eca7129a04a849c23a59cb

Reject-only analysis SHA-256: 00c9fe86d1490b6db8df3d08d31ad7f1cfb71d7ea015c969c8a6c35daa1c688c

Appendix D. Actual-parent receiver contrast

Table 8. Three fresh receiver trajectories in every row. Two selected USED parents from ONE exposed unique API; not independent tasks. fail/pass means pristine-fail/target-pass, a subset of invalid files. No nonfinal or unavailable cases. Bank counts cover two easy planted variants only for valid files.

Parent	Draft	Catch	Miss	Invalid	fail/pass	Bank
P1	none	0	0	3	2	0/0
P1	actual wrong	3	0	0	0	6/6
P1	authored repair	2	0	1	0	4/4
P2	none	0	0	3	1	0/0
P2	actual wrong	1	1	1	1	4/4
P2	authored repair	3	0	0	0	6/6

Qualified origin and explicit authored repair

Both parents originate in old no-draft model conversations. Frozen initial prompts contain null; original request, response, intent, tool and final source identities agree. Both whole files pass target and fail pristine. Three P1 expected lists and one P2 expected list are minimally corrected: 3208 to 3233 bytes and 4476 to 4481 bytes. All other bytes and AST nodes remain; corrected controls pass pristine and catch target. Repairs are authored, and their lengths and list AST shapes differ. Eight main plus eight independent native gate runs qualify the pair.

Native witnesses and counterevidence

ec-2-actual-wrong-1 retains P2's wrong body: identity-key input [1,'A','a',2] is expected as [1,'A',2]. Its final passes target and fails pristine. Body retention is observable syntax, not identification of internal copying. All three P1 actual-wrong receivers contain at least one body matching a repaired counterpart and catch the target; this is not a claim that every original test was repaired.

ec-1-authored-repair-3 retains two repaired bodies but adds a wrong check: for [1,'1',1,'1'] under key = str, it expects [1,'1'] instead of [1]. Thus even a qualified correct input bundle does not certify a generated whole file. Null controls themselves contain three pristine-fail/target-pass files. All 20 easy-bank opportunities are caught, including the valid miss of the difficult target.

Prospective execution and cost

Scientific and bounded-execution protocols were frozen before the 18 new conversations. Actual received coding POSTs: 41; provider-reported tokens: 203,364 (145,263 prompt; 58,101 completion). Every POST had a fresh included-quota check below 80; no HOLD or uncertain retry occurred. Generation performed 38 current-checkout native runs; three rejected actions did not execute. Original final analysis and the independent consumer each execute 56 native children and agree on all 18 outcomes, prompt blocks, body signatures and usage. Shared instrument/host/author remains a limitation.

The two old parents' generation calls are already in the iterator stratum and are not counted again. This fresh receiver stratum has its own completed exact-ancestor clean replay: seven workers, 128 native executions, five regenerated scientific outputs and 28 software checks. Earlier replay receipts are linked without repeating their workers. No receiver resampling follows these opposite parent effects. The contrast measures neither population harm nor covert collusive intent.

Actual-parent protocol SHA-256: 9204fb3c84cb322a715bad2ec0d4213cffdc11b00fde4420f8741d0a8b11f8ac

Actual-parent native analysis SHA-256: 5b8bf59fa83e61feaba8f100e6a8e20cfcfda7e203dd288b3e0c1b9f9cc043be