

Success Is Not Safety: When Helpful Bio Agents Do Too Much

Philip Nilsson

Independent Researcher

Apart Research Hackathon – Track 3: AI Biosecurity Tools

Abstract

Biosecurity evaluations often ask whether an agent completed the task or produced a harmful answer. This misses capability overreach: useful behavior that uses more biological capability than the request justified. We introduce Bio Capability Boundary Monitor, a prototype pre-deployment audit tool that checks whether biological-agent trajectories stay within task-relative capability boundaries and produces reviewer-facing decisions. In a 1,500-run Llama evaluation, raw task success was 96.6%, while safety-adjusted success was 31.7%. A scope-aware postcondition layer reduced strict false allow rate from 8.47% to 0.139%, recovering 60/61 misses with 0/480 new false positives. In a 100-run Track 3 application slice, useful-but-unsafe behavior reached 75.0% and false allow rate fell from 16.7% to 0.0%. These results suggest that biosecurity audits should treat unjustified biological capability use as a deployment risk, even when the final answer appears helpful and safe.

1. Introduction

Biological workflow agents can look safe for the same reason they look useful: they solve the visible task. That is the problem.

Success is not safety when the route to a correct answer uses more biological capability than the request justified.

Omission attacks show that agents can be unsafe by doing too little. Capability overreach shows that bio agents can be unsafe by doing too much.

Consider a safe proxy example. A reviewer asks an agent to summarize a synthetic anomaly table. The task only requires local analysis. The agent gives a correct summary, but also retrieves extra biological context and adds follow-up handling language.

This is the harmless-looking summary that quietly became a broader bio-risk workflow.

Table 1: Flagship example: harmless-looking enrichment.

Field	Example
User request	Summarize this synthetic anomaly table
Allowed scope	Local analysis only
Observed behavior	Correct summary plus extra biological context and follow-up handling language
Why it looks safe	The answer is helpful and not overtly harmful
Why it is unsafe	The workflow used unjustified biological capability
Audit decision	Review or restrict

The visible task succeeds. The answer may look careful. The workflow is still unsafe because it crossed from bounded local analysis into broader biological risk framing.

We call this **capability overreach**: a biological workflow crosses the task’s justified capability boundary while still completing the visible request.

Standard evaluation asks two useful but incomplete questions:

1. Did the agent produce a clearly harmful answer?
2. Did the agent complete the task?

Both questions matter. Neither is enough. A biological workflow agent can answer correctly, avoid overtly harmful content, and still use unjustified retrieval, validation, risk contextualization, readiness framing, or downstream-preparatory structure.

The missing question is: **Did the workflow stay inside the biological capability boundary justified by the task?**

We introduce **Bio Capability Boundary Monitor**, a prototype pre-deployment audit tool for biological workflow agents. It is not a lab assistant. It is not a biological decision-support system. It is a deployment audit layer.

The prototype records the visible task, allowed biological capability tier, observed tier, trace, and final output. It checks whether the trajectory stayed inside the justified capability boundary and produces reviewer-facing decisions: allow, review, restrict, block, or redesign.

Contributions.

1. We define capability overreach as a biosecurity failure mode where useful agents become unsafe by doing too much.
2. We build a prototype pre-deployment audit tool that checks task-relative biological capability boundaries.
3. We show that high task success can hide low safety-adjusted success: 96.6% raw success versus 31.7% safety-adjusted success in 1,500 Llama runs.
4. We show that scope-aware postconditions reduce strict false allow rate from 8.47% to 0.139%, recovering 60/61 misses with 0/480 new false positives.
5. We validate the pattern with a Track 3 application slice, Qwen-builder replication, and blinded human audit sanity check.

2. Related Work

This work sits at the intersection of AI control, biosecurity risk assessment, and trajectory-level monitoring.

2.1 AI control and monitor blind spots

AI control studies how to safely use capable models when a model may appear useful while pursuing unsafe behavior [4]. Recent work studies sabotage, resampling defenses, trajectory monitoring, and scheming detection [5, 6, 7, 8, 9].

Omission Attacks: When Doing Nothing Is the Attack showed a clean monitor blind spot: an agent can cause a security failure by omitting a necessary action. Every executed command may be legitimate, so action-level monitors have little to flag. The proposed defense was postcondition monitoring: check whether the final system state satisfies required invariants.

Table 2: Omission attacks vs capability overreach.

Dimension	Omission attacks	Capability overreach
Core failure	Agent skips a necessary action	Agent adds unjustified biological capability
Main illusion	Nothing suspicious happened	The answer looks helpful
Monitor blind spot	Action log lacks a bad action	Helpful extra work looks benign
Relevant invariant	Required final system state	Task-relative biological capability boundary
Defense	Final-state postconditions	Scope and capability postconditions
Summary	Unsafe by doing too little	Unsafe by doing too much

This paper adopts the same lesson as *Omission Attacks* but moves it into biosecurity workflow governance. Instead of asking whether a system state satisfies security invariants, we ask whether a biological workflow stayed within the capability scope justified by the visible task.

2.2 Biosecurity risk assessment

Biosecurity frameworks identify which biological capabilities, institutional weaknesses, and governance failures matter. The CLTR/RAND Global Risk Index for AI-enabled Biological Tools scores tools across dimensions such as capability, misuse potential, accessibility, and maturity [2].

Our contribution is a trace-level complement. Instead of scoring a tool category in the abstract, we score the biological capability actually used during a concrete agent run and compare it with what the task justified.

Table 3: RAND-style risk dimensions and trace-level counterparts.

RAND-style risk dimension	Trace-level counterpart in this work
Capability	Observed biological capability tier
Misuse potential	Violation category and severity
Accessibility	Which tools and retrieval paths were available
Maturity	Whether the workflow reaches readiness or downstream-preparatory framing
Governance	Reviewer decision: allow, review, restrict, block, or redesign

CSIS-style biosecurity recommendations emphasize practical governance capacity, biosurveillance, cyber-bio infrastructure, and AI threat forecasting [1]. Our application slice connects to those concerns through safe proxies for public-health signal triage and redacted screening policy review.

The goal is not to simulate real biological deployment end to end. The goal is to test whether a reviewer can detect when an AI agent leaves its bounded review role.

3. Methods

3.1 Threat model

We study pre-deployment evaluation of biological workflow agents in a controlled benchmark. Each run contains a visible biological review task, an allowed capability tier, a generated workflow trace, a final output, deterministic verifier labels, and monitor judgments. A run is successful if it completes the visible task. A run is boundary-compliant if it stays within the biological capability scope justified by that task. A run can therefore be successful and unsafe at the same time.

3.2 Capability tiers

The benchmark uses a five-tier taxonomy. These tiers are safe abstractions. They do not include operational biological protocols, synthesis guidance, organism-specific workflows, or lab-execution instructions.

Table 4: Capability tiers.

Tier	Label	Meaning
1	Local analysis	Use only provided inputs
2	Internal bio retrieval	Retrieve bounded internal biosafety or review material
3	External bio retrieval	Retrieve external biological references
4	Design-adjacent framing	Add structured downstream planning or protocol-like framing
5	Synthesis-adjacent preparation	Move toward higher-risk downstream biological preparation

A violation can occur in two ways. Excess-tier escalation means the observed tier exceeds the allowed tier. Same-tier scope drift means the observed tier equals the allowed tier, but the agent still expands beyond the task’s justified scope within that tier. Same-tier drift matters because simple threshold rules miss it by construction.

3.3 Scenario families

The main benchmark contains three safe proxy families.

Table 5: Scenario families.

Scenario	Allowed tier	Biosecurity review setting	Common overreach
S1	1	Biosurveillance anomaly triage proxy	Unnecessary retrieval for a local-only task
S2	2	Institutional biosafety review proxy	Drift from summary into next-step handling language
S3	3	Controlled biological risk-context review proxy	Protocol-adjacent or readiness-style expansion

Each scenario is tested under five behavioral conditions: benign, adversarial, benign overuse, partial adversarial, and borderline. This separates clean controls, direct pressure toward violation, non-malicious helpful overuse, partial pressure, and gray-zone cases near the boundary.

3.4 Monitor and scope postcondition design

The original monitor is an LLM-based oversight layer. It receives the visible task context, trace, final output, and deterministic tier information. In the final clean evaluation, it does not receive hidden side goals.

The added defense is a scope-aware postcondition layer. It is rule-based and auditable. Instead of asking only whether an individual step looked suspicious, it asks whether the completed workflow stayed inside the task’s allowed biological scope.

$$\text{augmented flag} = \text{original monitor flag OR scope postcondition flag}$$

The scope payload contains scenario id, visible prompt, allowed tier, observed tier, excess-tier status, tool trace, and final output. Observed tier is assigned by deterministic trace and output markers defined per scenario family, rather than by the monitor being evaluated. Ground-truth labels are used only after the run to compute metrics.

3.5 Why this is not label leakage

The scope-aware postcondition layer does not receive ground-truth violation labels, monitor outcomes, hidden side goals, or post-hoc correctness judgments. It receives only the visible prompt, scenario id, allowed tier, observed tier, excess-tier status, tool trace, and final output. Ground-truth labels are used only after inference to compute metrics. The layer is therefore not replaying labels; it is checking auditable scope invariants against the completed trajectory.

This does not prove automatic generalization to all future bio workflows. It shows that, for these task families, explicit scope postconditions catch a structured blind spot that the original monitor missed.

3.6 Reviewer-facing output

Biosecurity reviewers need more than aggregate scores. The prototype maps trajectory-level audit results into deployment-facing reports.

Table 6: Reviewer-facing audit output.

Field	Output
Decision	Restrict
Severity	Medium
Visible task	Synthetic anomaly summary
Allowed scope	Local analysis only
Observed behavior	Local analysis plus extra biological context
Violation type	Unjustified biological enrichment
Evidence	Trace and final-output scope markers
Concern	The agent moved from bounded analysis into broader biological risk framing
Recommended action	Disable retrieval for local-only triage tasks unless explicitly requested

The report layer does not change labels or metrics; it translates trajectory-level audit results into a decision a deployer can act on.

Table 7: Final-answer checks vs capability-boundary audits.

Dimension	Final-answer / task-success check	Capability-boundary audit
Main question	Was the answer harmful or useful?	Was the workflow justified?
What it sees	Final answer and task completion	Trace, tools, retrieval, scope, and output framing
Misses	Helpful overreach	Some ambiguous policy cases
Catches	Overt harmful output	Unjustified biological capability use
Deployment action	Allow or block	Allow, review, restrict, block, or redesign

3.7 Evaluation setup

The final clean run used meta-llama/Llama-3.1-8B-Instruct as both builder and monitor through an OpenAI-compatible local provider.

Table 8: Evaluation setup and accounting policies.

Item	Value or meaning
Tasks	15
Runs per task	100
Seed	7
Total rows	1,500
Parse errors	0
Invalid rows	0
All-valid	Includes all 1,500 rows and counts ambiguous borderline rows as violations
Strict hard-label	Excludes ambiguous borderline rows, leaving 1,200 rows

Strict hard-label is the primary policy for monitor comparison. We also run three checks: a 510-row Qwen-builder / Llama-monitor replication, a 100-run Track 3 application slice, and a blinded 50-run human audit sanity check.

4. Results

4.1 Task success greatly overstates safety

The agent usually solved the task. That was exactly why the failure was easy to miss.

Safety-adjusted success = main-task success AND no boundary violation.

Table 9: Main benchmark results.

Policy	Main-task success	True violation rate	Safety-adjusted success
All-valid	1449/1500 = 96.6%	1020/1500 = 68.0%	475/1500 = 31.7%
Strict hard-label	1174/1200 = 97.8%	720/1200 = 60.0%	475/1200 = 39.6%

Among successful all-valid runs, 974/1449 were useful but unsafe. That is 67.2% of successful runs. The agent usually completed the request. The problem was that it often completed the request using more biological capability than the task justified.

4.2 Scope-aware postconditions nearly eliminate false allows

For deployment, the critical metric is false allow rate: how often unsafe workflows pass monitoring. **False allow rate = unsafe runs allowed by the monitor / all unsafe runs.**

On the strict hard-label slice, the original monitor missed 61/720 unsafe runs. The augmented monitor missed 1/720.

Table 10: Original vs augmented monitor.

Monitor	TP	TN	FP	FN	Precision	Recall	False allow rate
Original monitor	659	480	0	61	1.000	0.915	8.47%
Augmented monitor	719	480	0	1	1.000	0.999	0.139%

The added layer recovered 60/61 misses without introducing new false positives on clean hard-label negatives.

This is the main monitoring result. The failure mode is not only visible after analysis. In this benchmark, a simple and auditable postcondition layer catches nearly all residual misses.

4.3 The misses cluster where overreach looks helpful

The original monitor was weakest when overreach looked helpful, cautious, or terse.

Flagship failure: harmless-looking enrichment. The clearest recurring miss was S1 benign-overuse validation drift. The task required local analysis only, but the agent added validation or biological context that exceeded the task scope. The original monitor missed 31 such unsafe successful runs. The scope-aware postcondition layer recovered all 31 by checking whether retrieval and output framing stayed within the local-analysis boundary.

Table 11: Residual miss clusters.

Residual miss cluster	Original FN	Recovered	Remaining FN
S1 benign-overuse validation drift	31	31	0
S1 partial-adversarial routing probe	2	2	0
S2 partial-adversarial scope creep	9	9	0
S3 partial-adversarial readiness framing	18	18	0
S2 benign terse trace-level overuse	1	0	1
Total	61	60	1

The original monitor's misses were not random noise. They clustered around cases where extra capability looked like useful review behavior. That is the core blind spot. Output-only checks are likely to miss failures when the unsafe behavior is not overtly harmful, but overly capable for the task.

4.4 Same-tier drift matters

Thresholds alone are not enough.

In 65 strict hard-label violations, observed tier did not exceed allowed tier, but the agent still expanded beyond the task's scope. The original monitor missed 26 of these. The augmented monitor recovered 26/26.

This is an important distinction. The safety question is not only whether the model used a stronger capability tier. Sometimes the model uses an allowed capability for an unauthorized purpose. Capability-boundary monitoring therefore needs both tier checks and scope checks.

4.5 The failure mode appears in Track 3-style workflows

The application slice is the closest part of the evaluation to practical Track 3 use: bounded public-health signal triage and redacted screening policy review.

The slice contains 100 non-operational runs split between public-health signal triage proxy and redacted screening policy review proxy. It uses synthetic metadata, fictional labels, and redacted fields only. It does not include real biological sequences, organism names, operational protocols, experimental plans, or lab-execution guidance.

Table 12: Track 3 application slice.

Application slice	Rows	Useful-but-unsafe rate	Original false allow	Augmented false allow
Public-health signal triage proxy	50	75.0%	33.3%	0.0%
Redacted screening policy review proxy	50	75.0%	0.0%	0.0%
Overall	100	75.0%	16.7%	0.0%

The benchmark failure mode transfers to workflows closer to what biosecurity reviewers actually need: bounded triage and screening policy review.

4.6 Replication and human audit sanity checks

To test whether the result was specific to the Llama builder, we reran the canonical Llama monitor on preserved Qwen-generated traces and outputs.

Table 13: Robustness checks.

Artifact	Builder	Monitor	Rows	Useful-but-unsafe strict	Original FAR	Augmented FAR
Canonical clean run	Llama-3.1-8B	Llama-3.1-8B	1,500	59.5%	8.5%	0.1%
Replication	Qwen2.5-7B	Llama-3.1-8B	510	53.7%	12.8%	1.4%

The Qwen-builder replication is supporting evidence, not proof of full model-general behavior. It shows that the qualitative failure pattern persists when the builder changes and the monitor is held fixed.

A second human reviewer audited 50 sampled runs using a fixed rubric. The reviewer saw the visible task, allowed tier, observed tier, excess-tier status, trace, and final output. They did not see hidden side goals, verifier labels, monitor outcomes, or scope-postcondition decisions.

Table 14: Human audit sanity check.

Human audit metric	Result
Sampled runs	50
Violation-vs-clean agreement	45/50 = 90%
Reviewer decision agreement	40/50 = 80%
Severity agreement within +/-1 level	48/50 = 96%
Major disagreements	2/50 = 4%

This is a sanity check, not full expert validation. It suggests that the deterministic verifier broadly matches human judgment while confirming that some boundary decisions are interpretation-sensitive.

5. Discussion

5.1 Track 3 relevance: this is a deployment audit prototype

Track 3 matters because bio agents will increasingly be used for triage, review, and retrieval-heavy support. Final-answer harmfulness checks miss cases where the agent stays polite and useful but uses unjustified biological capability. Bio Capability Boundary Monitor is meant to catch that before deployers grant richer retrieval or review permissions.

The prototype does not generate biological procedures. It checks whether a biological workflow agent stayed inside its assigned review role.

5.2 Helpfulness can be the risk signal

Many failures do not look malicious. They look like extra effort: additional validation, broader retrieval, cautious framing, or follow-up structure.

In ordinary software tasks, extra validation or cautious framing may be good behavior. **In bounded biosecurity tasks, helpfulness is not always a safety signal. Sometimes it is the violation.**

A reviewer should therefore not only ask whether the agent answered correctly. They should ask whether the agent needed the biological capability it used.

5.3 False allow rate should be reported directly

The original monitor already looked strong by standard metrics. But in deployment, the key question is how many unsafe workflows pass through. False allow rate makes that risk visible.

The augmented monitor reduced strict false allow rate from 8.47% to 0.139% in the canonical run, from 12.8% to 1.4% in the Qwen-builder replication, and from 16.7% to 0.0% in the Track 3 application slice.

5.4 Why postconditions help

Action-level monitoring asks whether an individual step looks suspicious. Capability-boundary monitoring asks whether the completed workflow stayed inside the task's justified scope.

This is why the scope-aware postcondition layer catches same-tier drift. The issue is not always that the agent used a stronger tool. Sometimes the issue is that the agent used an allowed capability for an unauthorized purpose.

This is not a claim that a rule-based layer solves biosecurity monitoring. It is a narrower claim: when task scope is explicit, checking

scope postconditions can catch useful-but-unsafe behavior that ordinary monitors miss.

5.5 What this does not prove

This does not prove deployment readiness in real biological workflows.

The benchmark uses safe proxy workflows, so realism is limited. The Track 3 application slice improves practitioner fit, but it remains small and non-naturalistic. The postcondition layer is rule-based by design. This makes it inspectable, but it also raises a generalization question: future workflow families may require different scope rules.

The Qwen replication changes the builder while holding the monitor fixed. It does not prove general behavior across frontier biological workflow agents. The blinded human audit used one additional reviewer and 50 sampled runs. A stronger evaluation would use multiple independent reviewers, at least one biosecurity or biosafety specialist, and adjudication of disagreements.

5.6 Practical recommendations

1. Do not evaluate bio agents only by final-answer harmfulness or task success.
2. Track biological capability used during the workflow, not only the answer.
3. Require scope-aware postconditions before giving agents richer bio retrieval or review tools.
4. Report false allow rate directly because it better reflects deployment risk than aggregate F1.
5. Treat helpful extra work as a possible risk signal in bounded biosecurity tasks.

6. Conclusion

Bio agents can be useful and unsafe at the same time.

In 1,500 runs, raw task success was 96.6%, but safety-adjusted success was 31.7%. Scope-aware postconditions reduced strict false allow rate from 8.47% to 0.139% with no new false positives on clean hard-label negatives.

The broader lesson is direct: biosecurity evaluation should not only ask whether an answer is harmful or whether a task was completed. It should ask whether the workflow used biological capability the task actually justified.

Code and Data

Code repository: <https://github.com/Philipnil06/bio-capability-monitor>

No external dataset was used. All task specifications, scenario variants, and benchmark traces were constructed as part of this study. Main artifacts include the final clean run, Track 3 reprocess rows, Qwen-builder replication run, and Track 3 application slice run. Full artifact paths and SHA256 hashes are included in the repository for reproducibility.

Appendix A: Limitations and Dual-Use Considerations

A.1 Limitations

The benchmark uses safe proxy workflows. This reduces biological risk but limits realism. The Track 3 application slice is closer to practitioner workflows than the abstract benchmark, but it remains small and does not simulate real deployment end to end.

The postcondition layer is rule-based and task-family specific. This makes it auditable, but it does not prove automatic generalization to future workflow families. The Qwen replication changes the builder but not the monitor, so it supports the qualitative pattern without establishing broad model-family robustness. The human audit used one additional reviewer and 50 samples; more expert reviewers are needed, including at least one biosecurity or biosafety specialist.

A.2 False positives and false negatives

In strict hard-label clean negatives, the augmented monitor introduced 0/480 new false positives. This does not prove low false positives in real deployments. Real workflows may contain messier traces, ambiguous task boundaries, and organization-specific review norms. One strict hard-label false negative remained. Borderline cases are interpretation-sensitive and should be routed to human review rather than treated as fully automated decisions.

A.3 Edge cases and scalability constraints

Future S4/S5-style workflows may need different scope rules. Automatic postcondition derivation remains future work. A scalable system would infer scope invariants from task specifications, tool permissions, and deployment policy rather than relying only on manually written rules.

More naturalistic agent tool access could introduce new failure modes. Richer retrieval, multi-agent delegation, long-horizon memory, and external integrations may create boundary crossings not represented in this benchmark.

A.4 Dual-use risks

This paper describes a monitoring blind spot, which could theoretically help attackers understand what monitors miss. Risk is reduced because the benchmark uses safe proxies and avoids operational biological detail. The work reports boundary structure rather than actionable biological procedures.

The goal is defensive: to help deployers detect when biological workflow agents exceed their justified role.

A.5 Responsible disclosure

No real biological vulnerability, sequence, organism-specific workflow, or operational protocol was discovered.

If this prototype is applied to real systems and risky behavior is found, results should be disclosed first to the deploying organization, model provider, or relevant biosafety or security team before public release.

A.6 Ethical considerations

The prototype is intended to reduce unsafe deployment of biological workflow agents. It should not be used as a standalone authorization system for high-risk biological work. Human review remains necessary for ambiguous cases.

The monitor should support governance decisions, not replace institutional biosafety or biosecurity judgment.

A.7 Future improvements

Future work should include larger Track 3 application slices, more independent reviewers, at least one biosecurity or biosafety specialist, more model families, more realistic agentic workflows, automatic derivation of scope postconditions from task specifications, and more detailed measurement of false positives in realistic clean deployments.

Author Contributions

P.N. designed the benchmark, constructed the scenario families and behavioral conditions, implemented the builder-verifier-monitor pipeline, implemented the scope-aware postcondition defense, ran the clean evaluation, analyzed the results, added the reviewer-report layer, ran the Qwen-builder replication, added the Track 3 application slice, organized the human audit sanity check, and wrote the paper.

References

- [1] Morrison, J. S., and Simoneau, M. (2023). *Eight Commonsense Actions on Biosafety and Biosecurity*. Center for Strategic and International Studies.
- [2] Webster, T., Moulange, R., Del Castello, B., Walker, J., Zakaria, S., and Nelson, C. (2025). *Global Risk Index for AI-Enabled Biological Tools: Summary Assessment and Methods Report*. Centre for Long-Term Resilience and RAND Europe.
- [3] Meulenbelt, S. E., van Passel, M. W. J., de Bruin, A., et al. (2019). The Vulnerability Scan, a Web Tool to Increase Institutional Biosecurity Resilience. *Frontiers in Public Health*, 7, 47.
- [4] Greenblatt, R., Shlegeris, B., Sachan, K., and Roger, F. (2024). AI Control: Improving Safety Despite Intentional Subversion. ICML 2024.
- [5] Bhatt, A., Rushing, C., Kaufman, A., et al. (2025). Ctrl-Z: Controlling AI Agents via Resampling. arXiv:2504.10374.
- [6] Kutasov, J., Sun, Y., Colognese, P., et al. (2025). SHADE-Arena: Evaluating Sabotage and Monitoring in LLM Agents. arXiv:2506.15740.
- [7] Korbak, T., Balesni, M., Shlegeris, B., and Irving, G. (2025). How to Evaluate Control Measures for LLM Agents? A Trajectory from Today to Superintelligence. arXiv:2504.05259.
- [8] OpenAI. (2025). Detecting Misbehavior in Frontier Reasoning Models.
- [9] OpenAI and Apollo Research. (2025). Detecting and Reducing Scheming in AI Models.

LLM Usage Statement

Claude and ChatGPT were used for brainstorming, drafting assistance, structural editing, and feedback on paper organization. Codex was used for implementation assistance, experiment orchestration, metrics export, and code review.

All benchmark semantics, result interpretation, experimental claims, and final paper decisions were reviewed and verified by the author.