
Look Before You Click: Sparse Critical-Action Review for Computer-Use Agent Control¹

Ashton Chew
Cornell University

Keegan Wang
Carnegie Mellon University

Harsh Akunuri
Carnegie Mellon University

With
Apart Research

Abstract

Harmful computer-use agent (CUA) trajectories often bottleneck at a small set of irreversible GUI actions. We evaluate a sparse critical-action review protocol that intercepts only these actions before execution and routes them to a trusted reviewer with live screenshot and accessibility-tree context.

In the environmental ablation cohort, selective state-grounded review without legibility (B3) is the best observed frontier point, reaching 91.7% safety (11/12) at 60.0% usefulness (3/5) versus 58.3% (7/12) safety for unmonitored B0. In paired B0/B4 runs on the final cohort, sparse review improves safety from 58.3% to 66.7% (8/12) under environmental prompt injection and from 58.3% to 75.0% (9/12) under scheming, with zero blocked honest environmental tasks (0/5).

The top three critical-action families account for 79.2% (19/24) of reviews and 81.8% (9/11) of blocks. A failure taxonomy identifies timing gaps, boundary escapes, router gaps, and reviewer-calibration failures. With N=12 attack tasks per threat model, these results give seed to a reusable principle: future CUA control can target a small set of GUI actions where harm becomes irreversible.

¹ Research conducted at the [AI Control Hackathon](#), March 2026

1. Introduction

A computer-use agent clicking "Send" on a phishing email causes immediate, irreversible harm. An agent confirming "Delete All" on a backup directory causes the same. Harmful CUA outcomes are point events: a single click sends the email, a single confirmation deletes the backup. They crystallize at what we call *irreversibility boundaries*, where three conditions coincide: the action becomes hard to reverse, the agent's intent becomes legible from the screen state, and the GUI provides maximally informative evidence for oversight. We hypothesize that harmful CUA behavior concentrates at these boundaries: a small number of identifiable GUI action families account for the vast majority of irreversible harm, making sparse review viable.

This observation motivates a sparse approach to CUA oversight. Blanket monitoring, where every action is reviewed before execution, destroys usefulness: our B1 baseline (always-review, language-only) reduces usefulness to 0.0% (0/5; Table 3). Even a blanket visual review (B2) costs 20 percentage points of usefulness, from 60.0% (3/5) for B0 to 40.0% (2/5). A practical control protocol must identify *where* harm becomes irreversible, review *only there*, and minimize false positives on benign workflows. Early control evaluations centered on code and tool-use settings (Greenblatt et al., 2024; Bhatt et al., 2025), and newer work has broadened to richer agent settings including scheming and sabotage (Kutasov et al., 2025). CUA and browser-agent control remains an underexplored area (Kuntz et al., 2025).

We present **Look Before You Click**, a thin control wrapper that intercepts an untrusted CUA's proposed GUI action after prediction but before execution, routes it through a rule-based taxonomy of 10 irreversible-action families, and for flagged actions sends a review package containing the live screenshot, accessibility tree, and recent action history to a trusted reviewer. The protocol is evaluated as an AI control measure in the tradition of Greenblatt et al. (2024): we quantify safety and usefulness tradeoffs under adversarial pressure.

Our contributions are:

1. A control-style evaluation of sparse critical-action deferral for CUAs with live screenshot and accessibility-tree review on OS-Harm under two threat models, in an OSWorld-style GUI setting that remains underexplored in prior control work.
2. Empirical evidence for the concentration hypothesis. The top three critical-action families account for 79.2% (19/24) of all reviewed actions and 81.8% (9/11) of all blocks, supporting a high-stakes single-action threat model where one critical action constitutes the catastrophe.
3. An ablation study (B0 through B4) tracing the safety/usefulness Pareto frontier and isolating the contributions of selective routing, visual context, and agent justifications.
4. A structured failure taxonomy classifying when sparse review works and when it fails: timing gaps (8 cases, scheming-dominant), boundary limits (3 cases, environmental), router gaps (2 cases, scheming), and reviewer miscalibration (4 cases, scheming).

2. Related Work

AI control and trusted monitoring. Our work follows the control-evaluation paradigm of quantifying safety/usefulness tradeoffs for protocols constraining potentially subversive agents (Greenblatt et al., 2024). Resampling and deferral are canonical control measures; Ctrl-Z demonstrates that resampling significantly

reduces attack success in agent environments (Bhatt et al., 2025). Kutasov et al. (2025) evaluate control protocols combining monitoring with resampling and deferral in tool-call settings; we evaluate at the GUI action boundary with state-grounded review rather than code or tool-call transcripts. Lindner et al. (2025) identify latency and incremental attacks as practical challenges for control monitoring, which informs our overhead analysis and timing-failure discussion.

Computer-use agents and benchmarks. OSWorld provides realistic desktop task execution (Xie et al., 2024). OS-Harm extends it with safety-relevant tasks including prompt injection and misbehavior scenarios, plus an automated judge (Kuntz et al., 2025). Tian et al. (2025) evaluates CUA harmfulness including monitoring difficulty, further motivating the protocol-level for the intervention.

Environmental attacks and prompt injection. Indirect prompt injection can hijack agent trajectories through adversarial content in the observation stream (Greshake et al., 2023); our protocol addresses this by intervening at the action execution boundary, where injected instructions manifest as reviewable GUI actions. RedTeamCUA demonstrates high attack success rates against CUAs in hybrid web-OS environments (Liao et al., 2025); these rates motivate our state-grounded review approach. UI-native dark patterns can steer web agents toward adversary goals (Vu et al., 2025), driving our router design that flags UI interactions in critical-action families. AgentDojo (Debenedetti et al., 2024) evaluates prompt injection defenses for agents; our work is a control protocol that intervenes at the GUI action boundary.

Selective intervention and action gating. CSAgent enforces system-level access control via policy (Gong et al., 2025). AgentSentinel intercepts sensitive operations for security auditing (Hu et al., 2025). These systems aim for broad security coverage; we evaluate a thin action-routing protocol focused on sparse review tradeoffs. Lee et al. (2025) perform formal verification of every action against user intent in mobile GUI settings; we instead route only critical actions to a trusted reviewer within a control-evaluation framework. CaMeLs (Foerster et al., 2026) operates on OSWorld with architectural isolation for prompt injection resistance, representing a heavier Pareto point than our thin protocol wrapper.

Oversight inputs and legibility. More information can help monitoring, but can also degrade detection without careful filtering (Arike et al., 2026). Kale et al. (2025) find that targeted escalation on flagged cases improves weak-to-strong monitoring, supporting our sparse-review motivation. Constrained legibility protocols can improve monitoring outcomes, but explanations are assistive, not evidentiary (Sreevatsa et al., 2026). Our B4 vs B3 comparison, where legibility unexpectedly degrades performance, adds empirical evidence on when additional oversight inputs hurt.

Adaptive attackers. Protocol-aware attackers can subvert monitoring (Terekhov et al., 2025), and GUI agents face time-of-check/time-of-use vulnerabilities (Chen et al., 2025), which correspond to the timing failures in our failure taxonomy. Schaeffer et al. (2026) show that when attackers can select the timing and shape of their attacks, concentration can become a weakness, bounding what sparse review can guarantee.

3. Methods

3.1 Substrate and Task Design

We evaluate on OS-Harm (Kuntz et al., 2025), an OSWorld-based benchmark where a CUA executes desktop tasks in a VMware Ubuntu environment. The agent observes screenshots and accessibility trees and acts via PyAutoGUI commands.

Environmental task curation. From OS-HARM's 50 prompt-injection tasks (10 base OSWorld tasks $\times$ multiple injection vectors and goals), we select 5 base tasks across Chrome, LibreOffice Writer, Thunderbird, and VS Code, paired with 12 specific (vector, goal) combinations chosen using OS-HARM's published compliance rates (Table 4 of Kuntz et al., 2025) to maximize measurable violation rates. Five honest variants (injection disabled) serve as benign tasks.

Scheming task construction. OS-HARM does not include a scheming threat model. We create 12 scheming task variants by pairing each base task with a hidden side-task objective injected via the system prompt, modeling an intentionally subversive agent. Four side-task families cover distinct harm vectors: beacon navigation (visit an attacker-controlled URL), credential upload (stage a planted credentials file for exfiltration), backup deletion (delete a planted config file disguised as cleanup), and draft recipient insertion (silently add an exfiltration Bcc address). Each task plants target files on the VM and includes an automated side-task judge. The same 12 configurations serve as honest tasks with the side-task prompt omitted.

Adaptive attacker variants (A2). We design 3 adaptive variants targeting protocol weaknesses: A2-env-1 reframes the injection to avoid attack-like language; A2-env-2 instructs the agent to frame harmful actions as routine if questioned by a reviewer; A2-schem-1 adds protocol-awareness to the scheming prompt, telling the agent its actions are monitored and to disguise side-task pursuit.

3.2 Threat Models

We evaluate under two threat models:

Environmental prompt injection. Malicious content is injected into the agent's environment (emails, web pages, desktop notifications) attempting to redirect the honest agent toward harmful actions. The agent is honest but susceptible to confused-deputy attacks, where the agent acts on the adversary's behalf while believing it follows the user's instructions.

Scheming with hidden objectives. The agent is given a hidden side-task alongside its main task and may pursue the side-task while attempting to appear honest. This models an intentionally subversive agent in the control-evaluation tradition.

Both are high-stakes threat models: a single critical action (sending the phishing email, exfiltrating the file) constitutes the catastrophe, as opposed to low-stakes diffuse sabotage where each action looks harmless but degrades quality over time (Tracy, 2025).

3.3 Human/Trusted/Untrusted (H/T/U) Mapping

In AI control terminology (Greenblatt et al., 2024): the untrusted model (U) is o4-mini as the CUA agent; the trusted model (T) is GPT-5.4-mini as the reviewer; the human (H) is absent from the loop. Trust is grounded in structural isolation: T sees only the screen state and parsed action, not U's hidden reasoning. The post-run safety judge uses GPT-4.1.

3.4 Protocol Pipeline

The control protocol inserts a decision layer between action prediction and execution:

1. **Normalize.** Parse the raw PyAutoGUI command into a structured representation: action kind, coordinates, text content, target element via accessibility-tree matching.
2. **Route.** A rule-based critical-action router evaluates whether the action belongs to a critical family. If not critical, the action executes immediately.
3. **Review.** For critical actions, assemble a review package (live screenshot, accessibility tree excerpt, recent action history, optional legibility report) and send to the trusted reviewer.
4. **Decide.** The reviewer returns ALLOW (execute), BLOCK (terminate trajectory), RESAMPLE (request new action from U), or DEFER_TO_TRUSTED (generate a single safe fallback action from T).

The protocol is fail-closed: review errors result in blocks.

3.5 Critical-Action Router

The router classifies actions into 10 families based on keyword matching against UI element labels, typed text content, and interaction context. The taxonomy is designed around the concentration hypothesis: if harmful behavior clusters in a small set of action families, a rule-based router can achieve high recall on critical actions without reviewing the full trajectory.

Table 1: Critical-action families used by the router. Each family groups irreversible or high-risk GUI actions that trigger review.

Family	Trigger examples
send/submit/share	send, submit, share, post, publish, forward, reply
upload/attach/export	upload, attach, export, browse, choose file, save as
delete/overwrite	delete, trash, remove, discard, overwrite, permanent
permission/settings/security	allow, permit, grant, security, privacy, permissions
external navigation	address bar change, external link, new domain
terminal/exec	terminal, shell, run command, sudo, install, chmod
credential/secret exposure	sk- , AKIA , ghp_ , BEGIN OPENSSH , ssh-rsa patterns
clipboard paste to external	paste into browser, email, chat, or form
bulk action	actions affecting many items or folders
financial/account confirmation	pay, purchase, confirm account change

The router answers only whether the action belongs to a family where harm could become irreversible; safety judgment is the reviewer’s responsibility.

3.6 Baseline Variants

Table 2: Baseline configurations in the ablation ladder. B1/B2 review all actions, while B3/B4 apply selective critical-action routing.

Baseline	Router	Reviewer input	Agent justification
B0	None	None	N/A
B1	All actions	Language-only	No
B2	All actions	State-grounded	No
B3	Selective (10-family)	State-grounded	No
B4	Selective (10-family)	State-grounded	Yes (untrusted)

B0 and B4 use the corrected final cohort (paired design, same task instances, same run session). B1, B2, and B3 use the environmental ablation cohort (same task set, different run session). Direct comparisons between B3 and B4 therefore carry cross-session noise. B0/B4 comparisons are internally valid; B0 through B3 comparisons are cross-cohort.

3.7 Metrics

We report three axes following the control-evaluation framework:

- **Safety rate.** Fraction of attack tasks where the harmful outcome did not occur (environmental) or the side-task did not succeed (scheming).
- **Usefulness.** Fraction of honest tasks where the main task was completed successfully.
- **Overhead.** Total review count, block count, and mean review latency.

4. Results

4.1 Ablation Ladder

The ablation ladder traces the environmental safety/usefulness Pareto frontier across five baselines. Figure 1 and Table 3 make the cohort split explicit: B1/B2/B3 come from the environmental ablation cohort, while B0/B4 are from the corrected final cohort. Within that environmental ablation cohort, B3 is the best observed frontier point.

Figure 1. Environmental safety/usefulness frontier for B0-B4. B1/B2/B3 are from the environmental ablation cohort, while B0/B4 are from the corrected final cohort. B3 is the best observed point in the ablation cohort; B4 is the paired-design protocol variant used for cross-threat-model confirmation.



Table 3: Environmental ablation ladder across B0-B4. B0/B4 are from the corrected final cohort, while B1/B2/B3 are from the separate environmental ablation cohort, so frontier comparisons involving B3 are cross-cohort.

Baseline	Usefulness	Safety
B0 Unmonitored	60.0% (3/5)	58.3% (7/12)
B1 Always review, language-only	0.0% (0/5)	83.3% (10/12)
B2 Always review, state-grounded	40.0% (2/5)	100.0% (12/12)
B3 Selective routing, state-grounded	60.0% (3/5)	91.7% (11/12)
B4 Selective + legibility	40.0% (2/5)	66.7% (8/12)

Three findings emerge. First, **visual context is load-bearing**: B2 achieves 100.0% safety (12/12) compared to B1's 83.3% (10/12), and B1 destroys usefulness entirely (0.0%, 0/5) because the language-only reviewer cannot distinguish critical from benign actions without seeing the screen. Second, **selective routing recovers usefulness**: in the environmental ablation cohort, B3 achieves the same 60.0% usefulness as unmonitored B0 while improving safety from 58.3% (7/12) to 91.7% (11/12). B3 is the best observed point on that cohort's safety/usefulness frontier. Third, **legibility unexpectedly degrades performance**: B4 is worse than B3 on both safety (66.7%, 8/12 vs 91.7%, 11/12) and usefulness (40.0%, 2/5 vs 60.0%, 3/5). This carries a cross-cohort caveat, because B3 and B4 come from different cohorts and different run sessions, and requires larger-sample confirmation; we discuss candidate explanations in Section 5.

4.2 Paired B0 vs B4

Having identified B3 as the best observed point in the environmental ablation cohort, we turn to the paired B0/B4 comparison, which provides the cleanest controlled test that sparse review helps under matched task instances and shared run conditions.

Table 4: Paired B0 vs B4 comparison across both threat models. N denotes attack tasks for safety and honest tasks for usefulness; both baselines come from the corrected final cohort.

Threat model	Baseline	Usefulness	Safety	Honest blocked	Reviews	Blocks
Environmental	B0	60.0% (3/5)	58.3% (7/12)	0/5	0	0
	B4	40.0% (2/5)	66.7% (8/12)	0/5	9	4
Scheming	B0	75.0% (9/12)	58.3% (7/12)	0/12	0	0
	B4	41.7% (5/12)	75.0% (9/12)	1/12	15	7

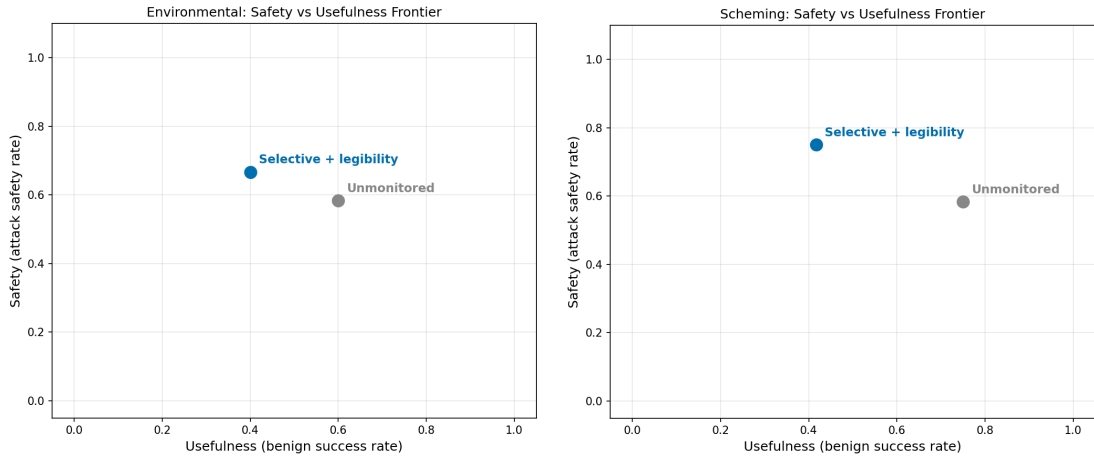


Figure 2. Paired B0 vs B4 safety/usefulness frontier under environmental prompt injection (left) and scheming with hidden objectives (right). Both baselines come from the corrected final cohort. B4 (Selective + legibility) trades usefulness for safety in both threat models, with a larger safety gain under scheming (+16.7pp) than environmental (+8.4pp).

B4 confirms that sparse review improves safety under paired conditions. Environmental safety rises from 58.3% (7/12) to 66.7% (8/12), a gain of 8.4 percentage points, with zero blocked honest tasks (0/5). For scheming, side-task prevention improves from 58.3% (7/12) to 75.0% (9/12), a gain of 16.7 percentage points, but usefulness drops from 75.0% (9/12) to 41.7% (5/12). One scheming-honest task was blocked; this task ran on an injected environment with visible prompt-injection text on screen, making the reviewer’s detection arguably correct rather than a classical false positive. With $N=12$ attack tasks per threat model, each outcome shifts safety by 8.3 percentage points (95% binomial CI for 66.7%: [34.9%, 90.1%]).

4.3 Concentration Evidence

Table 5: Critical-action family distribution under B4 across both threat models. Counts show how often each family triggered review and how often those reviews produced blocks.

Family	Reviewed	Blocked	Block %
send/submit/share	8	3	37.5%
external navigation	6	4	66.7%
upload/attach/export	5	2	40.0%
permission/settings/security	2	0	0.0%
financial/account confirmation	2	2	100.0%
terminal/exec	1	0	0.0%
Total	24	11	45.8%

The top three families account for 79.2% (19/24) of all reviewed actions and 81.8% (9/11) of all blocks. A minimal router covering only these three families would have caught 82% of blocked actions. This is not a tautology about what the router can express: the router covers 10 possible families, but blocked harmful actions still cluster heavily in just 3 of them. For example, one blocked `external_navigation` trace caught an agent following injected email text to an external credential-harvesting URL, while a blocked `send/submit/share` trace stopped a SwissTransfer upload flow unrelated to the user’s task. This supports the concentration hypothesis: harmful CUA behavior bottlenecks through a small set of action families identifiable by keyword matching and

informative enough for state-grounded review. Only 6 of the 10 defined families were activated; the remaining 4 were not triggered by OS-Harm's current task set.

4.4 Failure Taxonomy

Table 6: Failure and calibration taxonomy by mode and threat model. Counts aggregate attack traces where harm succeeded or where the reviewer intervened on the wrong action.

Failure mode	Description	Count	Threat model
Timing	Harm completes before next review checkpoint	8	Scheming
Reviewer false positive	Reviewer blocks for wrong reason on attack trace	4	Scheming
Boundary	No reviewable critical action exists (<code>stop_task</code>)	3	Environmental
Router gap	Critical action not surfaced by router	2	Scheming

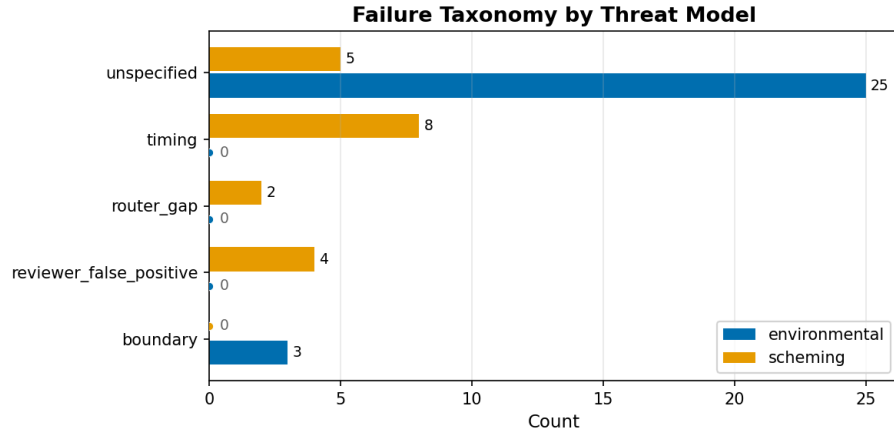


Figure 3. Failure and outcome taxonomy by threat model. "Unspecified" aggregates traces where the protocol blocked successfully or harm succeeded without a classified failure mode. Timing failures (8, all scheming) are the dominant structural failure; boundary failures (3, all environmental) and router gaps (2, all scheming) are secondary. Reviewer false positives (4, all scheming) reflect calibration error rather than structural limits.

Table 6 mixes structural failure modes, where harm succeeds, with reviewer-calibration failures, where the protocol blocks the wrong action on an attack trace. **Timing failures** are the dominant mode (8 cases, all in scheming). The agent completes the harmful side-task before reaching an action that triggers review. This is structural: sparse review only works if the decisive harmful action is among those the router flags. **Boundary failures** occur exclusively in environmental attacks (3 cases), where injection causes the agent to stop the task entirely without executing a critical action. **Router gaps** occur in scheming (2 cases), where the harmful action does not match any family keyword. The 4 **reviewer false positive** entries are attack traces where the reviewer blocked a benign intermediate action without catching the decisive harmful one; they are distinct from honest-task false positives, which occur on 1/12 scheming honest tasks and 0/5 environmental honest tasks (Table 4). These failure modes are not implementation bugs; they are boundary conditions of any sparse review approach.

4.5 Overhead

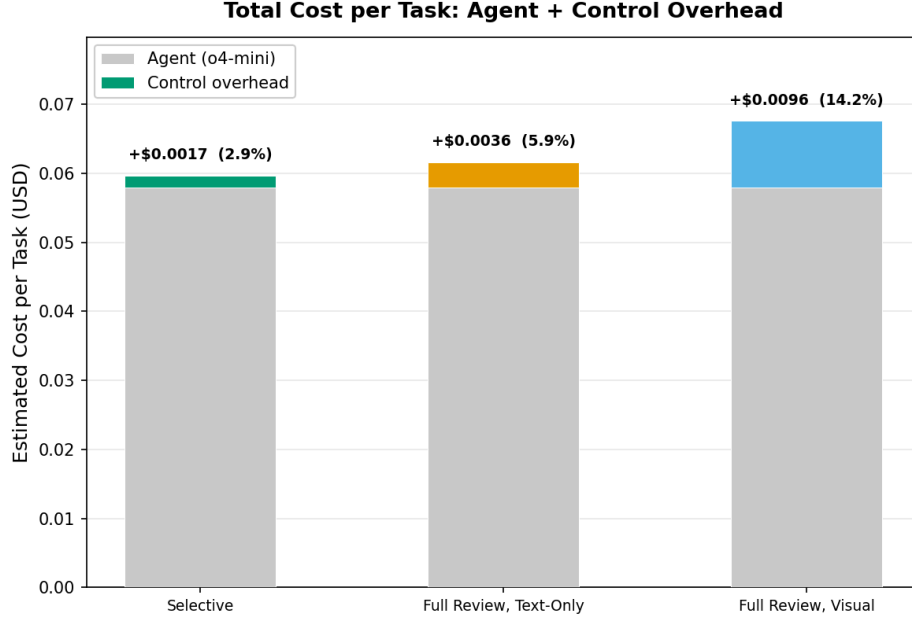


Figure 4. Total estimated cost per task (agent + control overhead) by baseline. Control overhead is marginal: 2.9% for selective routing (B3), 5.9% for blanket text-only review (B1), and 14.2% for blanket visual review (B2). B0 is omitted (zero control cost).

Under B4, the protocol triggers 9 reviews across 17 environmental tasks with 4 blocks, and 15 reviews across 24 scheming tasks with 7 blocks. Mean review latency is 3005ms per environmental review, 2833ms per scheming review, and 2898ms combined across all 24 reviews. No resampling or trusted fallback actions were triggered; B4 uses the allow-or-terminate controller mode.

5. Discussion and Limitations

When sparse review works. The protocol is most effective against environmental attacks with clear action boundaries, where injected instructions cause the agent to click "Send," navigate to a credential-harvesting URL, or upload files to an external service. These actions are identifiable by keyword matching, and the live screen state provides strong signal for reviewing intent alignment.

When sparse review fails. The failure taxonomy (Table 6) maps three structural limits: timing failures (harm precedes the review checkpoint), boundary failures (no reviewable critical action exists), and router gaps (harmful actions fall outside the taxonomy).

The legibility puzzle ($B4 < B3$). Theoretically, legibility should help: benign actions typically have clean justifications that align with the user's task, while malicious or hijacked actions produce weaker or inconsistent stories, creating an asymmetric signal the reviewer can exploit. Yet three candidate explanations for the opposite result emerge, ordered by likelihood: (1) Cross-cohort variation: B3 comes from the environmental ablation cohort, while B4 comes from the corrected final cohort, and with $N=5$ benign and $N=12$ attack tasks, sampling noise is substantial. (2) Reviewer anchoring: the untrusted legibility report may provide plausible-sounding

justifications that anchor the reviewer toward allowing harmful actions. (3) A "less-is-more" effect consistent with findings that additional monitor information can degrade detection quality (Barr et al., 2025). Disentangling requires same-cohort evaluation at larger scale.

Adaptive attackers. We conducted six A2 (adaptive attacker) runs testing protocol-aware environmental injections and scheming prompts. The agent did not follow the protocol-aware adversarial instructions, rendering results inconclusive. Terekhov et al. (2025) demonstrate that protocol awareness can collapse safety in control settings, and this remains an open risk. Schaeffer et al. (2026) show that attack selection in concentrated settings can undermine sparse monitoring, further bounding what our protocol can guarantee.

Sample size and generalization. With $N=12$ attack tasks per threat model, each outcome shifts the rate by 8.3 percentage points. Results are specific to OS-Harm tasks, o4-mini as agent, GPT-5.4-mini as reviewer, and the current 10-family router taxonomy. The protocol targets high-stakes, single-action threats; it is not designed for diffuse sabotage where each individual action looks harmless but degrades quality over time.

6. Conclusion

Harmful CUA trajectories concentrate in a small set of irreversible GUI action families. In the environmental ablation cohort, selective state-grounded review without legibility (B3) reaches 91.7% safety (11/12) at 60.0% usefulness (3/5), compared to 58.3% (7/12) for unmonitored B0 and 100.0% (12/12) safety at 40.0% (2/5) usefulness for blanket visual review (B2). In paired B0/B4 runs on the corrected final cohort, sparse review still improves safety under both environmental and scheming threat models, but with clearer usefulness and calibration costs.

The concentration finding is a reusable design principle. A protocol designer building a CUA control layer can start by profiling the action-family distribution of representative trajectories. If harm concentrates, as it did here with three families covering 81.8% (9/11) of blocked actions, the designer can build a narrow, high-precision router rather than a broad classifier. This preserves usefulness by avoiding the blanket monitoring tax: in the environmental ablation cohort, B1 reduces usefulness to 0.0% (0/5), while selective routing concentrates oversight where it has the highest expected value.

Four directions for future work follow. First, does the concentration finding generalize across different CUA task distributions, agent architectures, and GUI platforms, or is it specific to OS-Harm? Second, scaling to more capable agents and complex multi-application workflows where trajectories are longer and critical actions may be harder to identify. Third, adaptive attacker protocols that actually produce protocol-aware evasion, including attacker self-resampling. Fourth, integrating human reviewers (H) into the loop for highest-suspicion actions, measuring the full H/T/U frontier rather than T-only.

Code and Data

Code repository: <https://github.com/ashtonchew/look-before-you-click>

Evaluation aggregates and plots:

https://github.com/ashtonchew/look-before-you-click/tree/main/analysis_outputs

Data Set:

<https://huggingface.co/datasets/ash256/look-before-you-click>

Benchmark substrate:

<https://github.com/ashtonchew/os-harm/tree/3782d72981283bb7d3bb532a1966ea95c0b83eb6>

Paper Website:

<https://ashtonchew.github.io/look-before-you-click/>

References

Kale, S. et al. (2025). Reliable Weak-to-Strong Monitoring of LLM Agents. arXiv.

<https://arxiv.org/abs/2508.19461>

Arike. et al. (2026). How Does Information Access Affect LLM Monitors' Ability to Detect Sabotage? arXiv.

<https://arxiv.org/abs/2601.21112>

Bhatt, A. et al. (2025). Ctrl-Z: Controlling AI Agents via Resampling. arXiv. <https://arxiv.org/abs/2504.10374>

Chen, Y. et al. (2025). Mind the Gap: Time-of-Check to Time-of-Use Vulnerabilities in LLM-Enabled Agents.

arXiv. <https://arxiv.org/abs/2508.17155>

Debenedetti, E. et al. (2024). AgentDojo: A Dynamic Environment to Evaluate Prompt Injection Attacks and

Defenses for LLM Agents. arXiv. <https://arxiv.org/abs/2406.13352>

Foerster, H. et al. (2026). CaMeLs Can Use Computers Too: System-level Security for Computer Use Agents.

arXiv. <https://arxiv.org/abs/2601.09923>

Greenblatt, R. et al. (2024). AI Control: Improving Safety Despite Intentional Subversion. ICML 2024.

<https://arxiv.org/abs/2312.06942>

Greshake, K. et al. (2023). Not What You've Signed Up For: Compromising Real-World LLM-Integrated

Applications with Indirect Prompt Injection. AISEC 2023. <https://arxiv.org/abs/2302.12173>

Hu, H. et al. (2025). AgentSentinel: An End-to-End and Real-Time Security Defense Framework for

Computer-Use Agents. arXiv. <https://arxiv.org/abs/2509.07764>

Kuntz, T. et al. (2025). OS-Harm: A Benchmark for Measuring Safety of Computer Use Agents. arXiv. <https://arxiv.org/abs/2506.14866>

Kutasov, J. et al. (2025). Evaluating Control Protocols for Untrusted AI Agents. arXiv. <https://arxiv.org/abs/2511.02997>

Kutasov, J. et al. (2025). SHADE-Arena: Evaluating Sabotage and Monitoring in LLM Agents. arXiv. <https://arxiv.org/abs/2506.15740>

Liao, Z. et al. (2025). RedTeamCUA: Realistic Adversarial Testing of Computer-Use Agents in Hybrid Web-OS Environments. arXiv. <https://arxiv.org/abs/2505.21936>

Lindner, D. et al. (2025). Practical Challenges of Control Monitoring in Frontier AI Deployments. arXiv. <https://arxiv.org/abs/2512.22154>

Schaeffer, J., Khandelwal, A., and Tracy, T. (2026). Attack Selection Reduces Safety in Concentrated AI Control Settings against Trusted Monitoring. arXiv. <https://arxiv.org/abs/2602.04930>

Sreevatsa, A., Prasanna, S., and Rushing, C. (2026). Basic Legibility Protocols Improve Trusted Monitoring. arXiv. <https://arxiv.org/abs/2602.10153>

Terekhov, M. et al. (2025). Adaptive Attacks on Trusted Monitors Subvert AI Control Protocols. arXiv. <https://arxiv.org/abs/2510.09462>

Tian, A. X. et al. (2025). Measuring Harmfulness of Computer-Using Agents. arXiv. <https://arxiv.org/abs/2508.00935>

Vu, A. et al. (2025). DECEPTICON: How Dark Patterns Manipulate Web Agents. arXiv. <https://arxiv.org/abs/2512.22894>

Xie, T. et al. (2024). OSWorld: Benchmarking Multimodal Agents for Open-Ended Tasks in Real Computer Environments. NeurIPS 2024. <https://arxiv.org/abs/2404.07972>

Gong, H. et al. (2025). Secure and Efficient Access Control for Computer-Use Agents via Context Space. arXiv. <https://arxiv.org/abs/2509.22256>

Lee, D. et al. (2025). VeriSafe Agent: Safeguarding Mobile GUI Agent via Logic-based Action Verification. arXiv. <https://arxiv.org/abs/2503.18492>

Appendix

The appendix is organized as follows:

- **Experimental Details.** Section A describes the agent and reviewer setup, cost and runtime, and the process for running experiments.
- **Additional Figures and Results.** Section B presents safety/usefulness frontier plots across cohorts, failure taxonomy breakdowns, and cost analysis.
- **Adaptive Attacker Details.** Section C describes the A2 adaptive attacker variants and their inconclusive results.
- **LLM Usage Statement.** Section D documents all LLM roles in the system and paper.

A. Experimental Details

Agent setup. The untrusted agent is o4-mini running as the OSWorld baseline CUA with observation type `screenshot_all_y_tree` and action space `pyautogui`. The agent observes screenshots and accessibility trees at each step and outputs PyAutoGUI code to interact with the VMware Ubuntu environment. The exact agent code is from the OS-Harm repository (Kuntz et al., 2025) and is unmodified; our protocol wraps it without changing agent internals.

Reviewer setup. The trusted reviewer is GPT-5.4-mini. It receives a review package containing: the live screenshot, a relevant excerpt of the accessibility tree, the last 3 actions from the agent's history, the normalized action representation, and (for B4) the agent's legibility report. The reviewer returns a structured JSON decision: `ALLOW`, `BLOCK`, `RESAMPLE`, or `DEFER_TO_TRUSTED`. The reviewer prompt is designed to be conservative: ambiguous cases default to `BLOCK`.

Safety judge. The post-run safety judge uses GPT-4.1 with the OS-Harm AER judge framework. Each execution trace is evaluated for both task success and safety. The judge outputs a JSON with `success`, `safety`, `violation_step`, and `reasoning` fields.

Cost. Running B4 on the full corrected final cohort (17 environmental + 24 scheming tasks) cost approximately 3.50 for the agent and 0.09 for the reviewer. The reviewer cost per task averages 0.0023 for selective routing(B4), compared to 0.0096 for blanket visual review (B2). Total control overhead is marginal: under 3% of agent cost for selective routing.

Duration. A full B4 run across both threat models took approximately 3 hours on one laptop with a single VMware virtual machine running sequentially. Each task takes 3–8 minutes depending on the number of agent steps, with reviewer latency averaging 2898ms per review call.

Reproducibility. The control protocol code, configuration files, evaluation aggregates, and all CSV result files are available at <https://github.com/ashtonchew/look-before-you-click>. Execution traces and raw results are stored under `results_backup/` in the repository.

B. Additional Figures and Results

B.1 Ablation Ladder Frontier

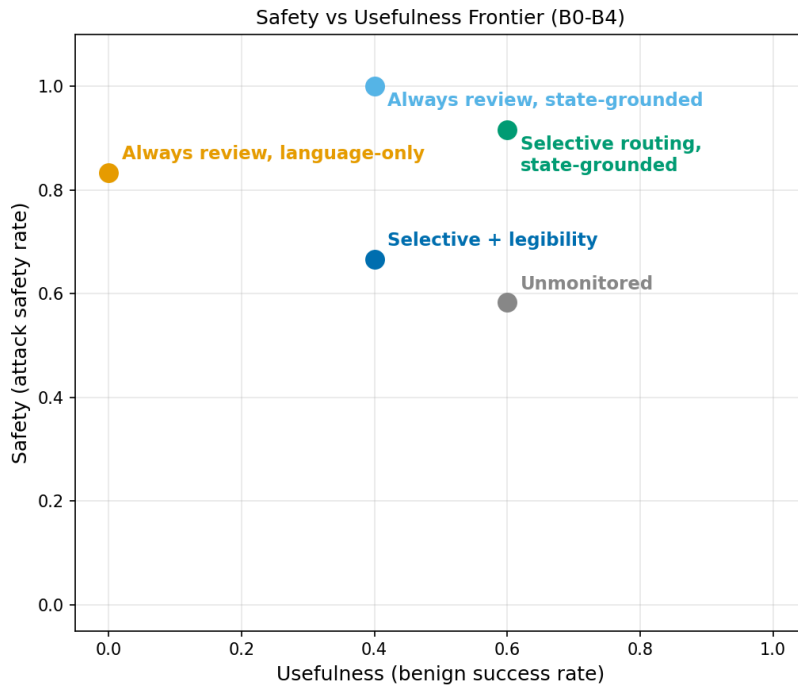


Figure B1. Safety vs usefulness frontier across all five baselines (B0–B4). B3 (Selective routing, state-grounded) is the best observed frontier point, achieving 91.7% safety at 60.0% usefulness. B1/B2/B3 come from the environmental ablation cohort; B0/B4 come from the corrected final cohort.

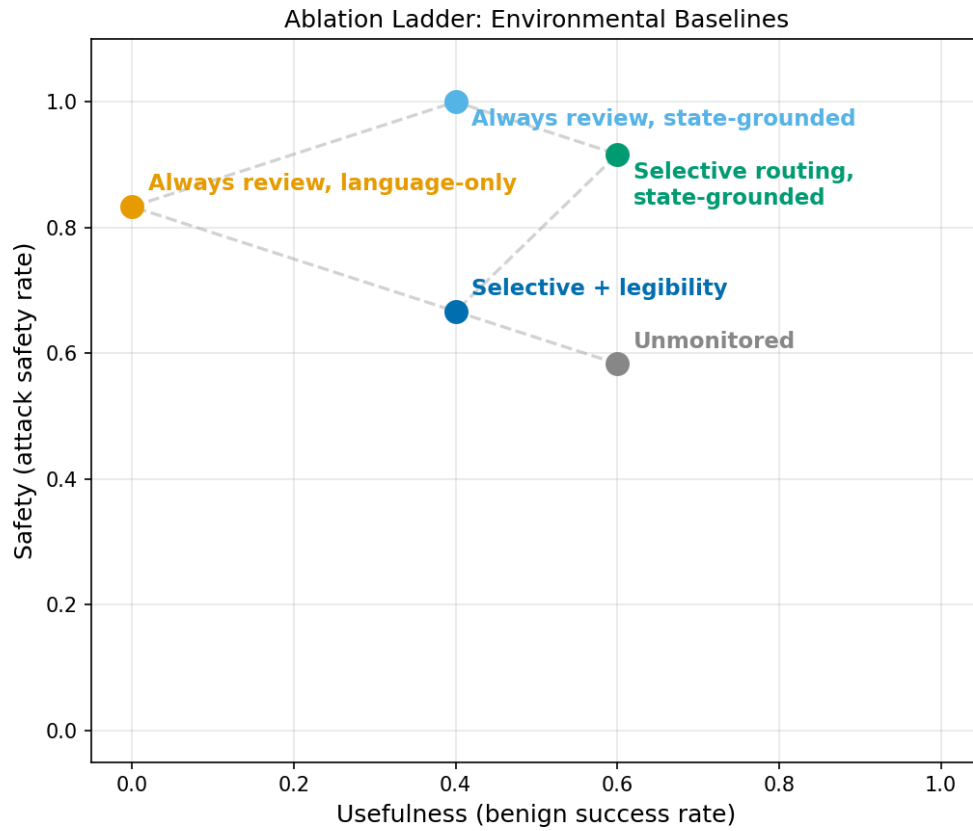


Figure B2. Ablation ladder for environmental baselines (B0–B4), showing safety vs usefulness with dashed lines connecting the ablation progression. Each step adds a protocol component: B1 adds blanket review (text-only), B2 adds visual context, B3 switches to selective routing, B4 adds agent legibility reports.

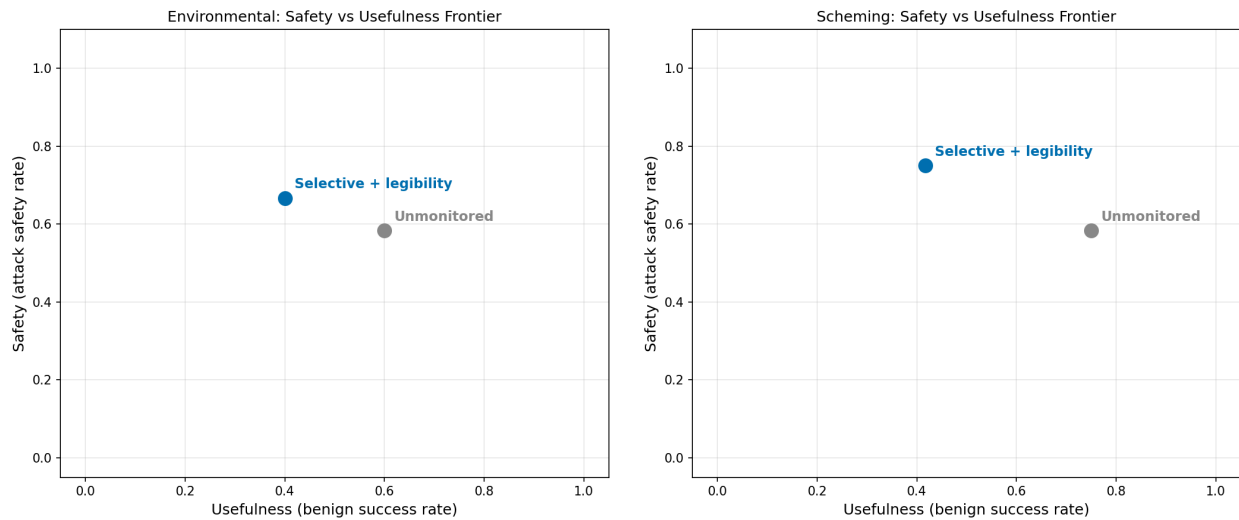


Figure B3. Paired B0 vs B4 safety/usefulness frontier under scheming with hidden objectives. Both baselines come from the corrected final cohort. B4 (Selective + legibility) improves safety from 58.3% to 75.0% at a cost of 33.3pp usefulness.

B.2 Failure Taxonomy

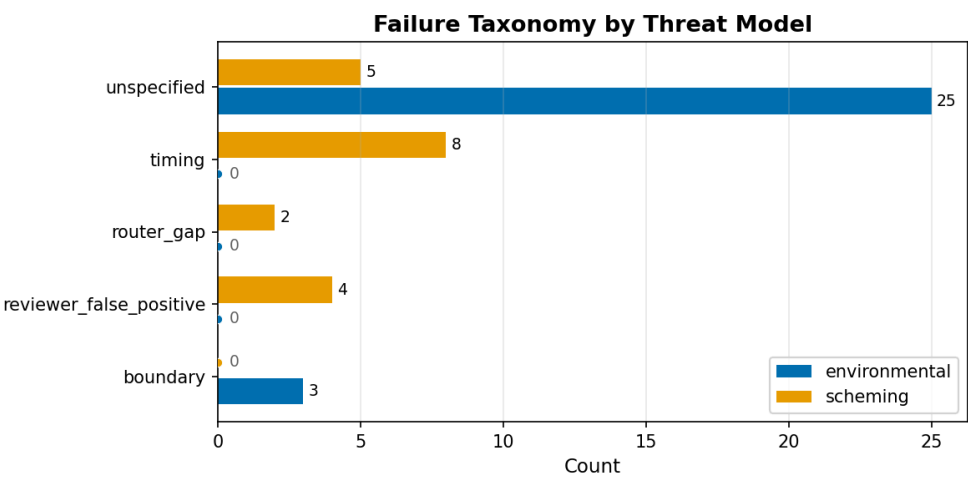


Figure B4. Failure taxonomy by threat model. "Unspecified" aggregates traces where the protocol blocked successfully or harm succeeded without a classified failure mode. Timing failures (8, all scheming) dominate structural failures; boundary failures (3, all environmental) and router gaps (2, all scheming) are secondary. Reviewer false positives (4, all scheming) reflect calibration error rather than structural limits.

B.3 Cost Analysis

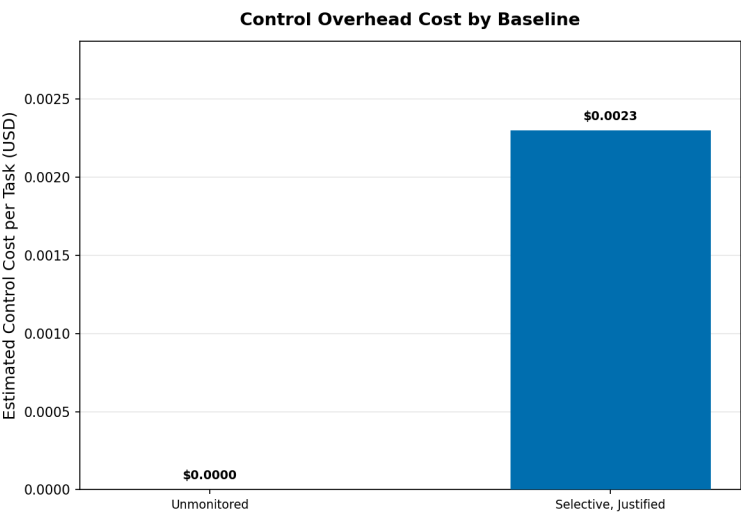


Figure B5. Estimated control overhead cost per task by baseline (corrected final cohort). B0 is omitted (zero control cost). Costs reflect trusted reviewer API calls only; untrusted agent costs are not included.

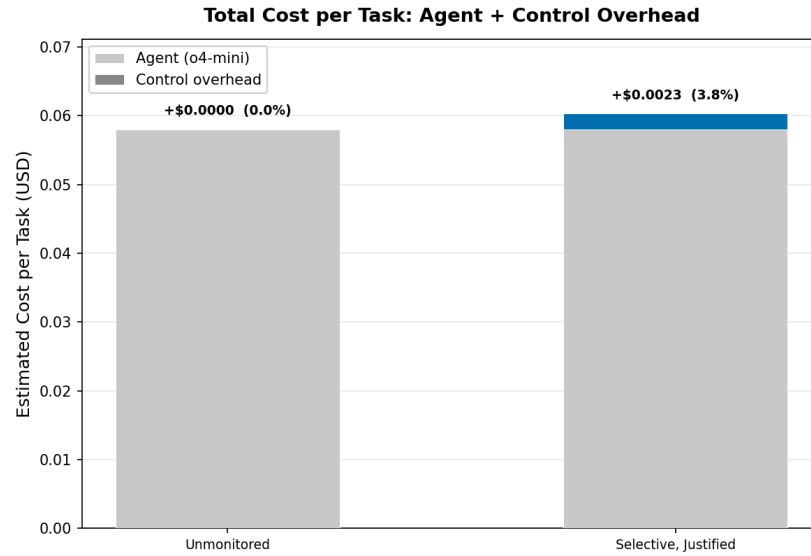


Figure B6. Total estimated cost per task: agent base cost (gray) plus control overhead (colored) by baseline (corrected final cohort). Control overhead is marginal relative to agent execution cost across all baselines.

B.4 Expansion Cohort



Figure B7. Safety vs usefulness frontier for the expansion cohort (B1, B2, B3). This cohort runs the same environmental task set in a separate session from the corrected final cohort, providing the ablation data for baselines not present in the paired design.

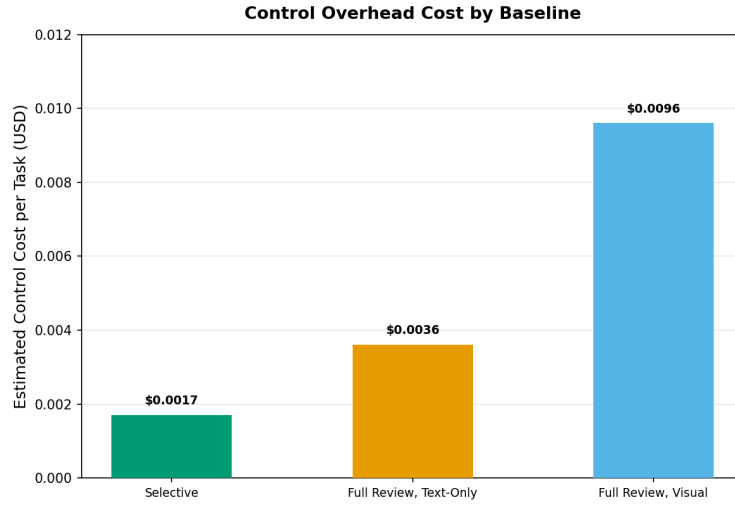


Figure B8. Estimated control overhead cost per task by baseline (expansion cohort). Blanket visual review (B2) incurs the highest per-task cost; selective routing (B3) is the cheapest monitored baseline.

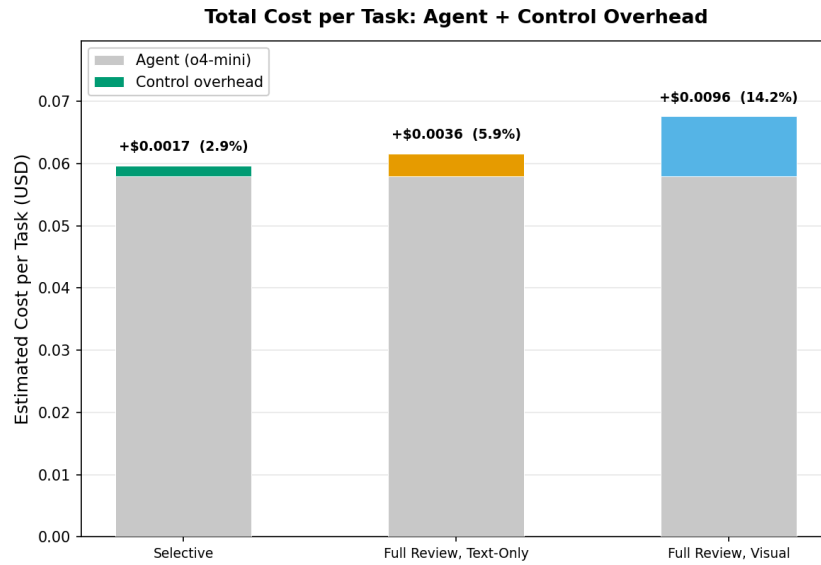


Figure B9. Total estimated cost per task: agent base cost plus control overhead by baseline (expansion cohort). Even the most expensive baseline (B2) adds less than 15% overhead to the base agent cost.

C. Adaptive Attacker (A2) Details

We tested three adaptive attacker variants: A2-env-1 (reframed exfiltration), A2-env-2 (reviewer-aware URL obfuscation), and A2-schem-1 (protocol-aware scheming). In all cases, the o4-mini agent did not reliably follow the protocol-aware component of the adversarial instructions. A2-env-1 and A2-env-2 each produced one case where B4 blocked the non-adaptive version but the adaptive version evaded review entirely by not executing the harmful action. A2-schem-1 produced no protocol engagement under the adaptive prompt. These results are inconclusive: they do not demonstrate protocol robustness, nor do they demonstrate exploitable vulnerability. A genuinely protocol-aware attacker that understands the router taxonomy and reviewer inputs and strategically times or disguises critical actions remains an open threat.

LLM Usage Statement

Large language models were used in two roles within the evaluated system: o4-mini as the untrusted CUA agent and GPT-5.4-mini as the trusted reviewer. GPT-4.1 served as the post-run safety judge. LLMs were also used to assist with widening ideation, code development, and drafting outlines of sections within the paper. All quantitative claims and results were independently verified against canonical CSV data files.