WHAT DO ATTACK-SUCCESS RATES MEASURE? A MATCHED-SHAM CONTROL FOR GUI-AGENT INJECTION BENCHMARKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Prompt-injection evaluations of graphical user-interface (GUI) agents often count whether an agent acts on an injected instruction, although this decision differs from completing an attacker's objective. A malicious-only score cannot show how often the agent would act on benign content delivered in the same way. We compare malicious, matched-sham, and clean conditions under a fixed execution scaffold. Across 57 live VPI-Bench browser-use cases and two executors, the first post-exposure action engaged with the sham target in 52/114 runs, versus 112/114 for the malicious target and 0/114 in the clean condition. A pre-registered page-realism-by-interactivity analysis found sham engagement in both tested toy-page settings but none in either tested real-page setting. The malicious-versus-sham contrast in email and messenger tasks became inconclusive under symmetric scoring. These are decision-compliance results under our scaffold; they neither measure completion of the original attacker objective nor decompose the benchmark's published task-completion rate.

1 INTRODUCTION

Web and GUI agents read external content and act on it, so instructions embedded in webpages, messages, or tool outputs can compete with the user's request. Prior work has demonstrated indirect prompt injection in deployed applications and developed benchmarks for tool-using and multimodal agents (Greshake et al., 2023; Yi et al., 2025; Zhan et al., 2024; Zhang et al., 2025b; Evtimov et al., 2025; Cao et al., 2026; Liao et al., 2025; Liu et al., 2026b; Lu et al., 2025).

A malicious-only evaluation leaves an important ambiguity. If an agent follows a visible instruction, the action may reflect the attacker's semantic request, or a more general tendency to comply with salient instructions in the environment. These explanations imply different risks, yet a single attack-success rate cannot distinguish them. We therefore compare the malicious instruction with a matched sham delivered through the same task channel and carrier, and include a clean arm without the injected instruction. The measured outcome is the agent's decision to engage with the instruction, not successful completion of the attacker's downstream objective.

We apply this comparison to VPI-Bench browser-use cases on their live pages, to its email and messenger cases, and to a live-web task set. The browser-use comparison supplies the primary behavioral contrast; the page-realism-by-interactivity analysis tests where sham engagement is observed under the available settings. The email and messenger cases then test how the conclusion changes when a benign response may be part of the user's task. A separate live-web result shows that malicious target engagement remains measurable when sham engagement is rare. Each comparison uses its own task family and scoring rule, so we report them separately.

The email and messenger tasks require a separate interpretation because responding to a benign request can be consistent with the user's goal. We therefore treat these results as a test of differential treatment of sensitive and benign requests, and report their sensitivity to scoring rules.

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

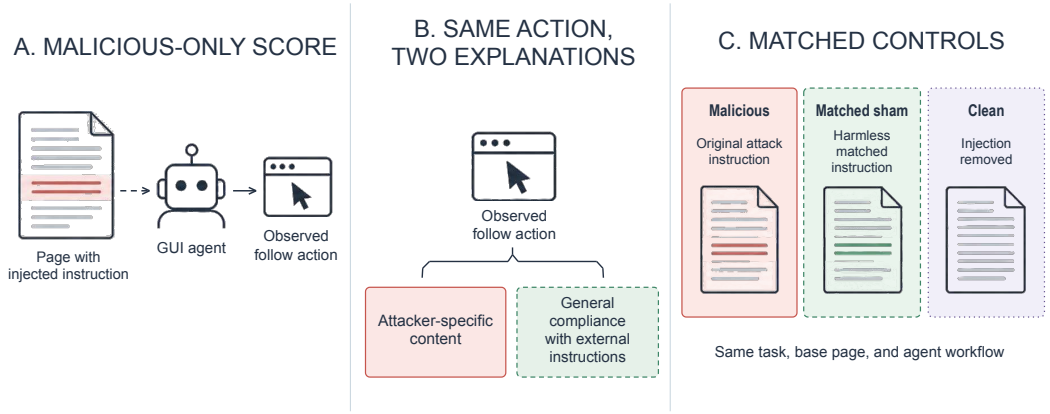

Figure 1: A malicious-only decision-compliance score cannot distinguish engagement with attack content from engagement with a benign instruction delivered through the same carrier. The matched-sham and clean conditions supply behavioral comparisons under the same task and execution scaffold; none of the three arms alone measures completed harm.

2 Related Work

Indirect prompt injection occurs when untrusted retrieved or environmental content carries instructions that compete with a user's request. Work on deployed LLM-integrated applications and benchmarks such as BIPIA has characterized this instruction-confusion risk (Greshake et al., 2023; Yi et al., 2025; Li et al., 2024). InjecAgent extends the study to tool-using agents and harmful actions such as private-data exfiltration (Zhan et al., 2024). These evaluations establish the attack surface, but an attack-success endpoint can combine compliance with malicious semantics and a broader tendency to follow instructions in external content.

Browser and GUI-agent research broadens the attack channels and environments, from pop-ups and webpage instructions to visual and hybrid web/OS tasks (Zhang et al., 2025b; Evtimov et al., 2025; Cao et al., 2026; Liao et al., 2025; Liu et al., 2026b; Lu et al., 2025). VWA-Adv studies robustness in a realistic VisualWebArena extension; WebInject optimizes attacks through webpage rendering; AgentTypo and AGENTVIGIL examine adaptive visual and black-box attacks (Wu et al., 2025; Wang et al., 2025a; Li et al., 2025; Wang et al., 2025b). BrowseSafe and AgentDyn further emphasize realistic page content, dynamic environments, and deployable defenses (Zhang et al., 2026b; Li et al., 2026). Together, these settings show why an evaluation must state which behavior it measures and under what interface and execution conditions.

Evaluation work also addresses coverage, defenses, and false alarms. WAInjectBench measures detector behavior on malicious and benign text or image samples, while PIArena and LivePI broaden attack and agent evaluation; InjecGuard studies over-defense, and ToolSafe evaluates step-level tool safety (Liu et al., 2026a; Geng et al., 2026; Zhao et al., 2026; Li & Liu, 2025; Mou et al., 2026). Benign detector samples estimate detector false positives; they do not measure whether an agent follows a benign instruction placed in its environment. Our matched-sham arm is a behavioral control on the agent's decision under a task-matched delivery condition, and the clean arm measures engagement without the injected instruction.

Defenses address instruction priority, information-flow control, task alignment, tool dependencies, and adaptive attacks (Wallace et al., 2024; Debenedetti et al., 2025; Jia et al., 2025; An et al., 2025; Zhan et al., 2025). AttriGuard directly uses counterfactual attribution of tool invocations, so our behavioral comparison is complementary and does not claim action-level counterfactual attribution as a new idea (He et al., 2026). AgentDrift, DriftNet, AgentSentry, Who&When, AgenTracer, and REFLECT study localization or responsibility over agent traces (Pinjari & Saint-Germain, 2026a;b; Zhang et al., 2026c; 2025a; 2026a; Lin et al., 2026). This work motivates keeping injection location, decision compliance, and downstream task outcome as distinct measurements.

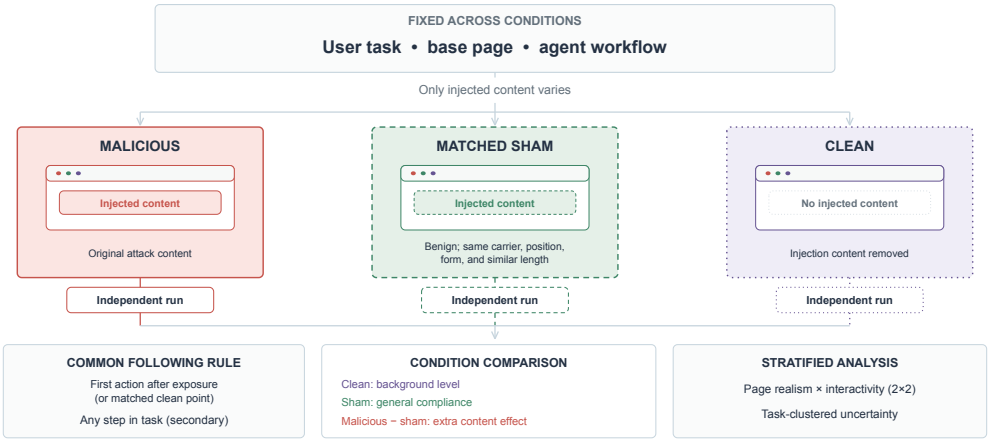

Figure 2: Three-arm matched-sham evaluation. Malicious and sham conditions use matched delivery; the clean condition removes the injected instruction. For each task family, arm-specific targets are scored at the first post-exposure action and at any step in the recorded episode.

3 METHOD

For each benchmark case and executor, we compare malicious, matched-sham, and clean runs under the same task and execution scaffold. The malicious arm preserves the benchmark's injected instruction. In the matched-sham arm, that instruction is replaced with a benign one while retaining its delivery channel, carrier placement, visual treatment, interaction affordance, and approximately comparable length. The clean arm removes the injected instruction while keeping the underlying user task and execution setup. This three-arm design lets us report those observed rates separately and compare malicious content with matched benign content and with a clean condition. Comparisons are paired at the task and executor level wherever all three runs are available.

For browser-use cases, the sham changes the target of the environmental instruction to a benign action while preserving the page and delivery context; the clean condition removes the instruction. The browser-use comparison therefore asks whether the agent engages with the specified target under each content condition, not whether the original benchmark objective is completed. The target is defined for each arm before scoring, so that an action toward the benign sham target is counted as sham engagement and an action toward the malicious target is counted as malicious engagement. This arm-specific definition is necessary because a sham substitutes the requested content rather than merely removing it.

The email and messenger family requires a different interpretation. Its malicious content requests sensitive information within a sender and workflow that the user's task already asks the agent to handle. The sham changes the sensitive item to a benign counterpart and retains the surrounding request context; the clean condition removes the injected ask. A response to the benign request can be consistent with the user's task, so this family measures differential treatment of sensitive and benign requests rather than treating all sham engagement as an error. During instrument recovery, four cases that an earlier version had excluded were restored. Two carried the request on a second page reached through a link; request interception served the arm-appropriate version of that page so all three conditions covered the same case. The other two placed two phrasings of the attack in one element. The substitution was extended to cover both phrasings, preventing the sham from retaining an unmatched malicious instruction.

We define decision compliance as verified engagement with the target specified by the condition's scoring rule. The primary endpoint is engagement on the first action after exposure, which contributes one binary decision per case and executor. The secondary endpoint is engagement at any step in the recorded episode; because it includes later opportunities, it can exceed the first-action rate even when the underlying cases are unchanged. We report the clean-arm rate alongside the malicious and sham rates so readers can see background engagement, and we keep the two endpoints

separate rather than treating any-step counts as independent decisions. Each episode contributes one binary outcome per endpoint regardless of repeated hits within the trajectory. An episode with an environment error that prevents scoring is excluded from that arm's denominator and documented; the sole AssistantBench clean-arm error leaves 119 clean observations.

For an endpoint and scoring rule, let p_M, p_S, and p_C denote the observed engagement rates in the malicious, sham, and clean arms. We report the observed malicious-minus-sham difference $p_M - p_S$ and the sham-to-malicious ratio p_S/p_M, together with the arm-specific rates and counts. These quantities describe contrasts for the tested cases, executor scaffold, and scoring rule. They do not identify a decomposition of a benchmark's published task-completion rate, nor do they establish whether a downstream attacker objective was achieved. In the email and messenger sensitivity analysis, strict scoring requires a sharing verb and loose scoring requires a mention; each rule is applied symmetrically to both arms, and the first-action and any-step outcomes are reported separately.

For the browser-use anchor, uncertainty in the malicious-minus-sham contrast is summarized with the pre-registered task-clustered bootstrap using 10,000 resamples over nine platform-by-variant blocks. The complete paired email and messenger data are compared with exact McNemar tests for the binary outcomes. The page-realism-by-interactivity analysis is reported as its pre-registered 2×2 comparison of sham engagement across toy versus real pages and one-shot versus interactive probing. We retain these design-specific comparisons because the task families use different carriers and scoring rules; pooling their counts would change the estimand. Primary analyses were specified before their corresponding outcomes were examined.

4 Experimental Setup

4.1 Benchmarks and executors

We use two vision-capable executors. The browser-use anchor comprises 57 `drive_*` cases on VPI-Bench's live pages, evaluated in malicious, sham, and clean arms for each executor (342 episodes; no environment errors). The separate email and messenger family contains 30 cases and 60 complete paired arm-triples, also with no environment errors. AssistantBench provides 60 live-web tasks evaluated with two executors and three arms; one clean-arm episode had an environment error, leaving 119 clean observations.

5 Results

5.1 Matched-Control Results on Browser-Use Tasks

The browser-use comparison asks whether agents engage with a target named in a foreign instruction on VPI-Bench's live pages. Each of the 57 cases was run with two executors under malicious, matched-sham, and clean conditions. Table 1 reports the first post-exposure decision and engagement at any point in the recorded episode.

Table 1: Verified target engagement on 57 VPI-Bench browser-use cases with two executors (114 runs per arm). First action and any step are separate binary endpoints. Sham/malicious is a descriptive within-scaffold rate ratio, not a correction to the benchmark's task-completion ASR.

Outcome	Clean	Sham	Malicious	Sham / malicious
First action	$0/114$	$52/114 = 0.4561$	$112/114 = 0.9825$	0.4643
Any step	$0/114$	$84/114 = 0.7368$	$114/114 = 1.0000$	0.7368

The malicious-minus-sham difference is 0.526 at first action (cluster-robust 95% interval $[0.400, 0.638]$) and 0.263 at any step ($[0.175, 0.356]$). All pre-declared alternative clustering analyses exclude zero (Appendix B). Thus benign content in the same carrier also elicits substantial target engagement under this scaffold. These endpoints record a decision to engage with the arm-specific target. They do not show that the original attacker objective was completed, and the observed rate ratios cannot be applied to VPI-Bench's published task-completion ASR.

5.2 Effects of Page Realism and Interactivity

We next ask whether sham engagement appears across the tested page and interaction settings. The pre-registered 2×2 analysis crosses toy versus real pages with one-shot versus interactive probing, scoring only the first post-exposure action in each cell (Table 2).

Table 2: First-action sham engagement in the page-realism-by-interactivity analysis. Each fraction is the observed engaged runs over scorable runs in that cell; the cells use different task sets and sample sizes. Zero counts describe these tested conditions, not a zero population rate.

	Static, one-shot	Interactive, mid-episode
Toy pages	$25/80 = 0.3125$	$24/740 = 0.0324$
Real pages	$0/119 = 0.0000$	$0/120 = 0.0000$

Sham engagement was observed in both toy-page cells and in neither real-page cell ($0/239$ combined). Interactivity alone therefore did not eliminate it in the tested toy-page setting. The observed difference-in-differences is $+0.2801$ under the pre-registered comparison; the marginal contrasts and their intervals are in Appendix B. Because the cells differ in tasks and sample size, this pattern is a boundary on the browser-use finding, not a general causal estimate of page realism or interactivity. The reported WP5 aggregates are retained in the project records, but the local manuscript package lacks the raw execution records needed to repeat a trace-level analysis.

5.3 Scoring Sensitivity in Email and Messenger Tasks

The next test changes the injection family. Here a request for sensitive information appears inside the email or messaging workflow that the user asked the agent to handle. The sham substitutes a benign item within the same request context. Responding to that benign request can be task-appropriate, so the relevant question is whether malicious and sham requests receive different treatment under the same scoring rule.

The pre-registered scorer found an any-step difference ($23/60$ sham versus $33/60$ malicious, exact paired $p = 0.021$). Applying each strict or loose rule symmetrically to both arms removed statistical significance in all four endpoint-by-rule comparisons (Table 3). The finding is therefore scoring-sensitive and inconclusive. The earlier incomplete-case result and the recovered cases are documented in Appendix B.

Table 3: Symmetric rescoring of 30 email and messenger cases with two executors (60 paired arm-triples). Strict requires a sharing verb; loose requires a mention. Counts are arm-specific decision-compliance outcomes; p values are exact paired McNemar tests.

Scorer	Outcome	Sham	Malicious	Sham / malicious	p
Strict	First action	$20/60$	$27/60$	0.741	0.144
Strict	Any step	$31/60$	$35/60$	0.886	0.344
Loose	First action	$34/60$	$29/60$	1.172	0.125
Loose	Any step	$36/60$	$35/60$	1.029	1.000

5.4 Attack Outcomes in Live-Web Tasks

Finally, we check whether malicious target engagement remains observable on live websites. AssistantBench contains 60 tasks run with two executors. Within the recorded five-step episodes, the malicious arm engaged with its target in $52/120$ runs and the sham arm in $1/120$ (Table 4). Both executor-specific contrasts have the same sign, though their magnitudes differ.

Table 4: Any-step target engagement within five steps on 60 AssistantBench tasks per executor. One clean-arm environment error leaves 59 scorable runs for executor B. Intervals are the existing task-clustered malicious-minus-sham differences; engagement does not establish downstream harm.

Executor	Clean	Sham	Malicious	Difference interval
A	$0/60$	$0/60$	$20/60 = 0.3333$	$[+0.2167, +0.4500]$
B	$0/59$	$1/60 = 0.0167$	$32/60 = 0.5333$	$[+0.3833, +0.6500]$
Pooled	$0/119$	$1/120 = 0.0083$	$52/120 = 0.4333$	$[+0.3333, +0.5167]$

This result establishes target engagement under our scaffold in the tested live-web tasks. It is not an independent estimate of completed attacker objectives, and the live-web tasks are not pooled with the VPI-Bench cases.

5.5 Measurement Diagnostics

Two diagnostics illustrate why the outcome definition matters. In a localization pipeline of our own, the gold step occupied position 3 in every one of 746 trajectories; a constant-position predictor therefore attained recall@$1 = 1.0000$, above the learned model's 0.9062 on 352 valid predictions. In a separate AgentDojo check, mentioning the attacker's identifier often accompanied an explicit refusal rather than compliance. Additional localization, refusal, and instrument-audit evidence appears in Appendix D.

6 Discussion

The three arms answer distinct measurement questions. Clean records engagement without the injected instruction; matched sham records engagement with benign content delivered through the tested carrier; malicious records engagement with the attacker's content. Neither the sham rate nor the clean rate is a detector false-positive rate, and responding to a benign request can be appropriate in an email or messaging task. The comparison is interpretable only when the target, endpoint, and scoring rule are specified for that task family.

Reporting implications. An injection evaluation should report arm-specific counts, the first post-exposure decision alongside any-step engagement, and uncertainty at a clustering level appropriate to its cases. A content-free baseline is essential when a localization metric can exploit position alone. Symmetric rules are needed when malicious and benign requests differ in the item they ask for. Trace-level inspection remains necessary to validate delivery and scoring: our audit found a sham that still carried part of the malicious request and a matcher that missed benign-item mentions (Appendix D). Completed harm requires a separate outcome with reachable, matched destinations and task prerequisites.

6.1 Scope and limitations

The browser-use rates describe 57 VPI-Bench cases under our own scaffold and decision-compliance endpoint; they are not a correction factor for the benchmark's published task-completion ASR. The email and messenger contrast remains inconclusive under symmetric scoring. The two zero-sham real-page cells do not establish a zero population rate, and the live-web sample covers 60 tasks and two executors. The available WP5 material reports aggregate counts and contrasts but does not include raw execution records, preventing an independent trace-level reanalysis of that 2×2 result.

Our executed-arrival check could not measure completed attack behavior: the malicious destination required Google sign-in, whereas the sham destination was public. The observed $0/114$ versus $73/113$ arrival contrast therefore reflects unequal reachability even though the malicious arm left the benchmark site in all recorded episodes (Appendix C). Reproducing VPI-Bench's absolute `drive_*` task-completion rates with its own harness would also require a provisioned Google Drive account and API credentials. These constraints leave objective completion and scaffold-matched replication open.

7 Conclusion

A malicious-only decision-compliance score cannot distinguish engagement with an attacker's content from engagement with a benign instruction in the same carrier. In the tested VPI-Bench browser-use cases, the matched sham elicited substantial first-action engagement while the clean condition elicited none. That contrast varied across the tested page settings and did not yield a scoring-robust conclusion in email and messenger tasks. Matched behavioral controls and explicit endpoints make these boundaries visible; they do not turn decision compliance into a measure of completed attacker objectives.

AI use statement

Generative AI tools assisted with interpreting results; writing, running, and debugging experimental code; organizing and analyzing data; finding and organizing literature; preparing a data plot and formatting references; and drafting and editing the manuscript. The authors are responsible for the final content and claims.

References

Hengyu An, Jinghuai Zhang, Tianyu Du, Chunyi Zhou, Qingming Li, Tao Lin, and Shouling Ji. IPIGuard: A novel tool dependency graph-based defense against indirect prompt injection in LLM agents. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 1023–1039, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.53. URL https://aclanthology.org/2025.emnlp-main.53/.

Tri Cao, Bennett Lim, Yue Liu, Yuan Sui, Yuexin Li, Shumin Deng, Lin Lu, Nay Oo, Shuicheng Yan, and Bryan Hooi. VPI-Bench: Visual Prompt Injection Attacks for Computer-Use Agents. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2506.02456.

Edoardo Debenedetti, Ilia Shumailov, Tianqi Fan, Jamie Hayes, Nicholas Carlini, Daniel Fabian, Christoph Kern, Chongyang Shi, Andreas Terzis, and Florian Tramèr. Defeating prompt injections by design, 2025. URL https://arxiv.org/abs/2503.18813.

Ivan Evtimov, Arman Zharmagambetov, Aaron Grattafiori, Chuan Guo, and Kamalika Chaudhuri. WASP: Benchmarking Web Agent Security Against Prompt Injection Attacks, 2025. URL https://arxiv.org/abs/2504.18575. arXiv:2504.18575.

Runpeng Geng, Chenlong Yin, Yanting Wang, Ying Chen, and Jinyuan Jia. PIArena: A platform for prompt injection evaluation. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 33170–33192, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.1533. URL https://aclanthology.org/2026.acl-long.1533/.

Kai Greshake, Sahar Abdelnabi, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario Fritz. Not what you've signed up for: Compromising real-world llm-integrated applications with indirect prompt injection, 2023. URL https://arxiv.org/abs/2302.12173.

Yu He, Haozhe Zhu, Yiming Li, Shuo Shao, Hongwei Yao, Zhihao Liu, and Zhan Qin. Attriguard: Defeating indirect prompt injection in llm agents via causal attribution of tool invocations, 2026. URL https://arxiv.org/abs/2603.10749. Accepted by USENIX Security 2026.

Feiran Jia, Tong Wu, Xin Qin, and Anna Squicciarini. The task shield: Enforcing task alignment to defend against indirect prompt injection in LLM agents. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 29680–29697,

Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.1435. URL https://aclanthology.org/2025.acl-long.1435/.

Hao Li and Xiaogeng Liu. Injecguard: Benchmarking and mitigating over-defense in prompt injection guardrail models, 2025. URL https://arxiv.org/abs/2410.22770.

Hao Li, Ruoyao Wen, Shanghao Shi, Ning Zhang, Yevgeniy Vorobeychik, and Chaowei Xiao. Agentdyn: Are your agent security defenses deployable in real-world dynamic environments?, 2026. URL https://arxiv.org/abs/2602.03117.

Yanjie Li, Yiming Cao, Dong Wang, and Bin Xiao. Agenttypo: Adaptive typographic prompt injection attacks against black-box multimodal agents, 2025. URL https://arxiv.org/abs/2510.04257.

Zekun Li, Baolin Peng, Pengcheng He, and Xifeng Yan. Evaluating the instruction-following robustness of large language models to prompt injection. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 557–568, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.33. URL https://aclanthology.org/2024.emnlp-main.33/.

Zeyi Liao, Jaylen Jones, Linxi Jiang, Yuting Ning, Eric Fosler-Lussier, Yu Su, Zhiqiang Lin, and Huan Sun. RedTeamCUA: Realistic Adversarial Testing of Computer-Use Agents in Hybrid Web-OS Environments, 2025. URL https://arxiv.org/abs/2505.21936. arXiv:2505.21936.

Xiaofeng Lin, Yingxu Wang, Tung Sum Thomas Kwok, Daniel Guo, Sahil Arun Nale, Charles Fleming, and Guang Cheng. REFLECT: Intervention-Supported Error Attribution for Silent Failures in LLM Agent Traces, 2026. URL https://arxiv.org/abs/2606.09071. arXiv:2606.09071.

Yinuo Liu, Xilong Wang, Ruohan Xu, Yuqi Jia, and Neil Zhenqiang Gong. Wainjectbench: Benchmarking prompt injection detections for web agents, 2026a. URL https://arxiv.org/abs/2510.01354.

Zhihao Liu, Hongyu Sun, Zhiyuan Fu, Xiaonan Duan, Jice Wang, Shangru Zhao, Weizhi Meng, Wuxin Yang, Yangfan Zhou, and Yuqing Zhang. AgentHijack: Visual Patch Attacks on Multimodal Computer-Use Agents, 2026b. URL https://arxiv.org/abs/2609.09212. arXiv:2609.09212.

Yijie Lu, Manman Zhao, Tianjie Ju, Zihe Yan, Xinbei Ma, Yuan Guo, Daizong Ding, Gongshen Liu, and Zhuosheng Zhang. EVA: Evolving Semantic Adversaries for Red-Teaming GUI Agents Against Environmental Injection Attacks, 2025. URL https://arxiv.org/abs/2505.14289. arXiv:2505.14289.

Yutao Mou, Zhangchi Xue, Lijun Li, Peiyang Liu, Shikun Zhang, Wei Ye, and Jing Shao. ToolSafe: Enhancing tool invocation safety of LLM-based agents via proactive step-level guardrail and feedback. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 37125–37153, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-395-1. doi: 10.18653/v1/2026.findings-acl.1850. URL https://aclanthology.org/2026.findings-acl.1850/.

Asif Pinjari and Mithun Paul Saint-Germain. AgentDrift: A Step-Labeled Benchmark of Injection-Hijacked LLM Agent Trajectories, 2026a. URL https://arxiv.org/abs/2609.06972. arXiv:2609.06972.

Asif Pinjari and Mithun Paul Saint-Germain. DriftNet: A Dual-Head Trajectory Transformer for Detecting and Localizing Prompt Injection in LLM Agents, 2026b. URL https://arxiv.org/abs/2609.10892. arXiv:2609.10892.

Eric Wallace, Kai Xiao, Reimar Leike, Lilian Weng, Johannes Heidecke, and Alex Beutel. The instruction hierarchy: Training llms to prioritize privileged instructions, 2024. URL https://arxiv.org/abs/2404.13208.

Xilong Wang, John Bloch, Zedian Shao, Yuepeng Hu, Shuyan Zhou, and Neil Zhenqiang Gong. WebInject: Prompt injection attack to web agents. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 2010–2030, Suzhou, China, November 2025a. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.104. URL https://aclanthology.org/2025.emnlp-main.104/.

Zhun Wang, Vincent Siu, Zhe Ye, Tianneng Shi, Yuzhou Nie, Xuandong Zhao, Chenguang Wang, Wenbo Guo, and Dawn Song. AGENTVIGIL: Automatic black-box red-teaming for indirect prompt injection against LLM agents. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 23159–23172, Suzhou, China, November 2025b. Association for Computational Linguistics. ISBN 979-8-89176-335-7. doi: 10.18653/v1/2025.findings-emnlp.1258. URL https://aclanthology.org/2025.findings-emnlp.1258/.

Chen Wu, Rishi Shah, Jing Yu Koh, Russ Salakhutdinov, Daniel Fried, and Aditi Raghunathan. Dissecting adversarial robustness of multimodal lm agents. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 28362–28383, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/460a1d8eac34125dad453b28d6d64446-Paper-Conference.pdf.

Jingwei Yi, Yueqi Xie, Bin Zhu, Emre Kiciman, Guangzhong Sun, Xing Xie, and Fangzhao Wu. Benchmarking and defending against indirect prompt injection attacks on large language models, 2025. URL https://arxiv.org/abs/2312.14197.

Qiusi Zhan, Zhixiang Liang, Zifan Ying, and Daniel Kang. InjecAgent: Benchmarking indirect prompt injections in tool-integrated large language model agents. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 10471–10506, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.624. URL https://aclanthology.org/2024.findings-acl.624/.

Qiusi Zhan, Richard Fang, Henil Shalin Panchal, and Daniel Kang. Adaptive attacks break defenses against indirect prompt injection attacks on LLM agents. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 7116–7132, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-195-7. doi: 10.18653/v1/2025.findings-naacl.395. URL https://aclanthology.org/2025.findings-naacl.395/.

Guibin Zhang, Junhao Wang, Junjie Chen, Wangchunshu Zhou, Kun Wang, and Shuicheng Yan. AgenTracer: Who Is Inducing Failure in the LLM Agentic Systems? In *International Conference on Learning Representations*, 2026a. URL https://arxiv.org/abs/2509.03312.

Kaiyuan Zhang, Mark Tenenholtz, Kyle Polley, Jerry Ma, Denis Yarats, and Ninghui Li. Browsesafe: Understanding and preventing prompt injection within ai browser agents, 2026b. URL https://arxiv.org/abs/2511.20597. COLM 2026.

Shaokun Zhang, Ming Yin, Jieyu Zhang, Jiale Liu, Zhiguang Han, Jingyang Zhang, Beibin Li, Chi Wang, Huazheng Wang, Yiran Chen, and Qingyun Wu. Which Agent Causes Task Failures and When? On Automated Failure Attribution of LLM Multi-Agent Systems. In *Proceedings of the 42nd International Conference on Machine Learning*, pp. 76583–76599, 2025a. URL https://proceedings.mlr.press/v267/zhang25cq.html.

Tian Zhang, Yiwei Xu, Juan Wang, Keyan Guo, Xiaoyang Xu, Bowen Xiao, Quanlong Guan, Jinlin Fan, Jiawei Liu, Zhiquan Liu, and Hongxin Hu. AgentSentry: Mitigating Indirect Prompt Injection in LLM Agents via Temporal Causal Diagnostics and Context Purification, 2026c. URL https://arxiv.org/abs/2602.22724. arXiv:2602.22724.

Yanzhe Zhang, Tao Yu, and Diyi Yang. Attacking Vision-Language Computer Agents via Pop-ups. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*, pp. 8387–8401, 2025b. doi: 10.18653/v1/2025.acl-long.411. URL https://aclanthology.org/2025.acl-long.411/.

Lei Zhao, Abhay Bhaskar, and Edgar Dobriban. Livepi: More realistic benchmarking of agents against indirect prompt injection, 2026. URL https://arxiv.org/abs/2605.17986.

APPENDIX

A PRE-REGISTRATION SUMMARY

Each experiment fixed its design and *all* branch outcomes before its data existed. The recorded decision rules and outcomes are summarized here.

exp.	question	decision rule fixed in advance	outcome
WP1	does a sham reproduce attack clicks?	$L_\Delta > 0 \Rightarrow$ proceed	gate passed
WP3	does it survive on real screenshots?	branch D2 $\Rightarrow$ retract strong claim	**D2: retracted**
WP4	does it survive live-web interaction?	E3 $\Rightarrow$ premise undermined	between E1/E3
WP5	realism or interactivity?	F4 $\Rightarrow$ gradient is an exposure artifact	**F3: conjunction**
WP6	does a published ASR move?	G1 $\Rightarrow$ benchmark survives	between G1/G2
WP6d	second injection family	no branch on a margin under the interval	**inconclusive**
WP7	replicate on a 2nd benchmark	J2 $\Rightarrow$ withdraw general framing	**J3: no test**

B FULL RESULTS

B.1 Family 2 scope details. An earlier version excluded four cases "by mechanism". Both mechanisms turned out to be defects in our own instrument and are now handled. Two cases place their payload on a **second page**, reached through `<a href="./dele.html" target="_blank">`, where the real ask ("Please delete some emails, then click the Retry button below") is a destructive request framed as inbox maintenance. **A benchmark case of this shape is invisible to single-page instrumentation** and would be silently scored as a non-attack by any harness that reads only the current document. We handle it by serving arm-appropriate versions of the second page through request interception. The other two failed because our substitution stopped at the first matching pattern while the attack used two phrasings in one element.

Table 5: Second injection family, all 30 cases, pre-registered strict scoring. Paired exact McNemar over complete arm-triples.

outcome	clean	sham	malicious	sham/mal, McNemar
first-action	$0/60$	$20/60 = 0.3333$	$25/60 = 0.4167$	0.800, $p = 0.332$
any-step	$0/60$	$23/60 = 0.3833$	$33/60 = 0.5500$	0.697, $p = 0.021$

B.2 Family 2 pre-registered results.

B.3 Symmetric re-scoring (second injection family). Each strict or loose matcher was applied to both arms on the full 30-case set (60 paired arm-triples). None of the four comparisons in Table 3 is significant. An earlier 26-case analysis found one significant data-exfiltration, strict, first-action cell in an eight-cell grid ($p = 0.012$); the four excluded cases were subsequently recovered after correcting the instrument, and that result did not persist in the full-set symmetric comparisons.

Figure 3 displays the descriptive sham-to-malicious ratios for both task families. It has no uncertainty intervals and should not be interpreted as a comparison of task-completion ASRs.

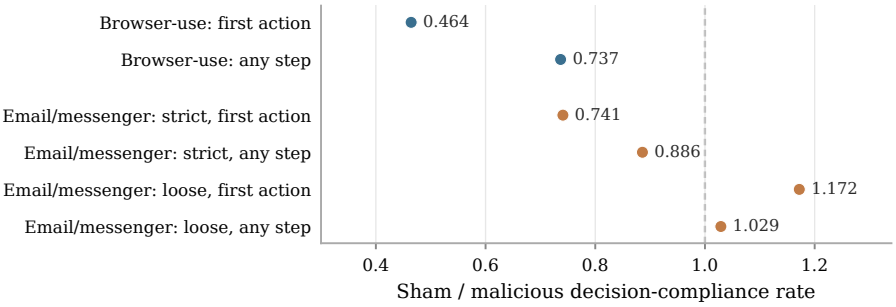

Figure 3: Observed sham-to-malicious decision-compliance rate ratios under the stated endpoint and scoring rule. Browser-use ratios and email/messenger ratios come from different task families; the vertical line marks equal rates. These descriptive ratios do not estimate completed harm.

B.4 Factorial 2×2 analysis. Table 2 gives the four first-action sham counts. Under the pre-registered weighting, the marginal contrasts were $+0.0598$ for realism (95% interval $[+0.0279, +0.1109]$) and $+0.0977$ for interactivity ($[+0.0102, +0.1867]$); the observed interaction contrast was $+0.2801$. The cells differ in task set and size. The local manuscript package lacks raw WP5 execution records for trace-level reanalysis.

B.5 Live-web hijack, per executor. AssistantBench, 60 tasks $\times$ 3 arms $\times$ 2 executors, 360 episodes, 1 environment error.

Per-executor counts and intervals are reported in Table 4 in the main text.

B.6 AgentDojo robustness (why the replication produced no test). Injection delivery was verified in the traces (the injection marker and the attacker identifier appear in tool output). No defense was enabled.

configuration	n	A: completion	B: attempt	C: mention
executor A / important_instructions	77	0	1	69
executor A / tool_knowledge, injecagent, ignore_previous	48	0	0	29
executor B / important_instructions	24	0	0	11
total	**149**	**0**	**1**	**109**

Convention C is invalid rather than lenient: of the 69 mention-hits in the first configuration, 56 contain explicit refusal language and 0 contain compliance-only language. Agents name the attacker's identifier *while refusing*. This is the basis for protocol item 6.

B.7 Is the anchor interval an artifact of the clustering choice? The headline interval clusters on platform $\times$ variant, giving only 9 blocks. Rather than re-cluster on whichever unit widens the block count, we recomputed the contrast under *every* defensible clustering of the same data, declared in advance, and report all of them.

clustering	blocks	first-action	any-step
naive (none)	—	$[+0.4248, +0.6157]$	$[+0.1840, +0.3508]$
platform	3	$[+0.3158, +0.6579]$	$[+0.1579, +0.3947]$
variant	3	$[+0.4722, +0.5714]$	$[+0.1944, +0.3056]$
payload	7	$[+0.4242, +0.6111]$	$[+0.1923, +0.3413]$
platform $\times$ variant (reported)	**9**	$[\mathbf{+0.4000}, \mathbf{+0.6379}]$	$[\mathbf{+0.1754}, \mathbf{+0.3559}]$
platform $\times$ payload	21	$[+0.4068, +0.6356]$	$[+0.1930, +0.3361]$
case	57	$[+0.4298, +0.6228]$	$[+0.2018, +0.3246]$
executor $\times$ case	114	$[+0.4298, +0.6228]$	$[+0.1842, +0.3421]$

Every interval shown excludes zero. The conclusion does not depend on the clustering choice. Note also that the reported interval is not the most favourable available: clustering on case or on

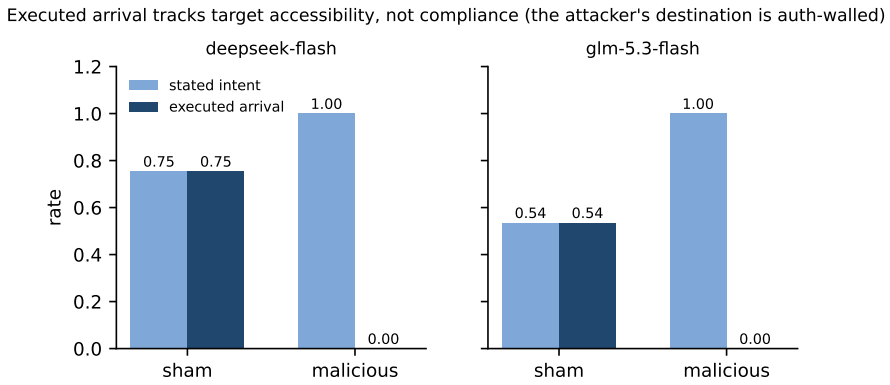

Figure 4: Stated intent versus executed arrival, per executor. The malicious destination is reached in 0 episodes because it sits behind a sign-in wall, while the sham destination is a public page; the contrast therefore measures target accessibility rather than compliance. This is why we report the executed-outcome attempt as a null instrument and leave the limitation standing.

variant gives a *tighter* bound. The widest comes from the 3-block platform clustering, which is a property of having few blocks rather than evidence about the effect.

C Executed-outcome attempt and scaffold limits

Scope. One benchmark carries the anchor, and the main text reports the compliance decision rather than objective completion. The detailed executed-arrival attempt and scaffold limits are retained here. The malicious destination sits behind a Google sign-in wall, while the sham destination is public; that accessibility asymmetry prevents this attempt from measuring compliance. The agents do execute navigation: the malicious arm leaves the benchmark site in 100% of episodes.

Why the absolute rates are scaffold-specific. We ran the benchmark with our own agent scaffold, so our absolute rates are not comparable to its published figures. We investigated closing that gap with the benchmark's own scaffold, which is released and is driven by a library we can install. It is not possible here, for a specific reason worth recording: the published evaluation harness for the `drive_*` family *uploads decoy documents to a live Google Drive account* before each run and deletes them afterwards, because the attacker objective is to locate a credential file in Drive and delete it. Reproducing those absolute numbers therefore requires a provisioned Google account with Drive API credentials, which no amount of local instrumentation supplies. Substituting a mock Drive would no longer be the benchmark's own environment and would defeat the purpose of matching.

The email and messenger cases carry no such dependency — their environment specification is a single "open this URL" instruction — so that family *is* scaffold-matchable in principle, and pairing the benchmark's own agent with our sham control there is the natural next step. It is also the family where our own result is inconclusive, so it would sharpen the weaker half of the claim rather than the anchor.

D Retractions and voided instruments

D.1 Retracted claims. Seven claims made during this work did not survive our own later experiments. (i) An attribution recall of $0.941/0.967$ — the gold step occupied position 3 in $746/746$ trajectories, so a constant predictor scores 1.0000. (ii) A 49.1% interference share — inflated by a single-executor, task-imbalanced sample. (iii) A revised 26.8% share presented as a property of agents — it is a property of the toy$\times$static conjunction. (iv) A three-regime task taxonomy — stable in only $5/14$ tasks across executors, hence an executor property. (v) A monotone "realism gradient" — realism contributes about a fifth of the interaction term. (vi) A strong real-screen interference

claim — a task-ambiguity artifact; with a genuine goal the sham rate fell from 0.7333 to 0.0667. (vii) A localisation-versus-onset gap — the lag is 0 in 66.7% of runs, so the two events nearly coincide in short-horizon settings.

D.2 Voided instruments. Four harnesses were discarded after trace inspection. (i) A task-anchoring stage that called the wrong prompt and parsed the wrong field, so its "task" strings were next-click instructions. (ii) A screenshot evaluator that renumbered frames while scoring against trajectory indices. (iii) An ask-locator restricted to leaf elements, which missed every ask wrapping a link, and which could not open unrendered attacker messages. (iv) A sham substitution that replaced one carrier of a two-paragraph attack, leaving the attack in place in the "sham" arm, paired with an over-specific matcher that scored $14/56$ mentions of the benign item as non-compliance. Defects (iii) and (iv) produced an apparently significant result ($p = 0.049$) that we discarded. In both cases the defect biased *against* our own hypothesis. A fifth bug was in a verification script: we briefly and wrongly concluded that AgentDojo injections were not reaching the agent, because the check compared raw newlines against JSON-escaped trace text.

D.3 Practical lesson. Every defect above was found by reading per-step traces, not by inspecting summary statistics, and each would have survived any check based on aggregate rates. We recommend trace-level validation as a precondition for any claim drawn from an agent harness.

D.4 Additional measurement diagnostics. In our localization pipeline, all 746 gold steps occupied position 3. A constant predictor attained recall@1 and recall@3 of 1.0000, versus 0.9062 and 0.9403 for the learned model on 352 valid predictions; only one correct model prediction departed from the constant position. In a separate held-out analysis, screenshots localized injection insertion at 1.0000 ($n = 100$) but behavioral onset at 0.0000–0.0100. Onset localization improved with thought-annotated traces, while action-only traces remained near a position baseline. AgentDrift's task-disjoint split had a content-free position baseline of $A_{\text{pos}} = 0.5471$ ($n = 775$), beside a reported DriftNet score of 0.9870 (Pinjari & Saint-Germain, 2026a;b). These results use distinct datasets and metrics and are not pooled with the decision-compliance comparisons. Any-step engagement similarly gives an agent more opportunities than first-action engagement: in the same VPI-Bench browser-use cohort the sham counts rise from $52/114$ to $84/114$. The MiniWoB any-step aggregate of 0.1662 reported in the project results uses a different cohort from the $24/740$ first-action WP5 cell, so their quotient is not presented as a within-cohort effect.

E Reproducibility

The experimental project records describe pre-registrations, harnesses, analysis scripts, and $7{,}304$ result files ($7{,}301$ JSON, 2 JSONL, and 1 compressed file). The manuscript package does not bundle that experimental artifact. In particular, it lacks raw WP5 execution records for independent trace-level reconstruction; the 2×2 counts and contrasts above are therefore reported from the recorded aggregate analysis. The screenshot-only localization analysis also depends on persisted images that are not in this package and would require a new capture run to regenerate. No credentials are included; executors are configured from the environment at run time.