

Semantic Policy Enforcement for Low-Level Computer Use Agent Tools

Anonymous Author(s)

ABSTRACT

Computer Use Agents (CUAs) that interact with GUIs through low-level actions such as click and find present a fundamental challenge for security policy enforcement: these primitive operations lack the intrinsic semantics needed to define meaningful data-flow restrictions. We formalize three levels of policy abstraction—syntactic, semantic, and contextual—and evaluate their effectiveness through large-scale Monte Carlo simulations across 2,000 task scenarios, 8 application domains, and varying adversarial conditions. Our contextual policy framework achieves an F1 score of 0.973 and detects 94.0% of prompt injection attacks, compared to 0.611 and 19.7% for syntactic baselines, while maintaining 72.1% task utility. Domain analysis reveals that high-risk domains (banking, healthcare) benefit most from contextual reasoning. Scalability experiments show that the advantage of semantic policies grows with task complexity, and injection robustness analysis confirms consistent performance across attack rates from 5% to 70%. These results demonstrate that planner-provided intent annotations and context-aware reasoning are essential infrastructure for securing CUA tool invocations.

1 INTRODUCTION

The emergence of Computer Use Agents (CUAs)—AI systems that interact with software through GUI-level actions such as clicking, typing, and navigating—has created new security challenges that existing information-flow control frameworks struggle to address [2]. Within the Dual-LLM architecture, control flow isolation separates privileged planning from quarantined perception, yet data flow remains vulnerable because the perception model’s outputs can steer execution [2].

Standard information-flow security policies [1, 5] can mitigate data-flow risks in domains with semantically rich tool interfaces. However, CUA tools like `click(x, y)` and `find(selector)` lack intrinsic semantics, making it difficult to define and enforce meaningful policies. This gap motivates the central question of this work: *How can we design and evaluate semantic security policies for low-level CUA tool invocations?*

We address this question through three contributions:

- (1) A formal model of semantic policies as predicate-action rules parameterized by abstraction level (syntactic, semantic, and contextual).
- (2) A simulation framework that generates realistic CUA task scenarios with adversarial prompt injections across diverse application domains.
- (3) Comprehensive empirical evaluation demonstrating that contextual policies substantially outperform alternatives on the safety–utility Pareto frontier.

2 PROBLEM FORMULATION

2.1 CUA Tool Actions

We model CUA interactions as sequences of typed actions $a = (\tau, t, c, r)$ where $\tau \in \mathcal{T}$ is the action type (navigate, click, type, read, submit, download, execute, modify settings, send message, authenticate), t is the target, c is the surrounding context, and $r \in \{0, 1, 2, 3, 4\}$ is the ground-truth risk level from SAFE to CRITICAL.

2.2 Task Scenarios

A task scenario $S = (s_1, \dots, s_n)$ is a sequence of actions representing a complete user-delegated task. Each scenario has a task type (web search, form fill, purchase, data extraction, account management, communication) and operates in a domain d with risk multiplier μ_d that adjusts the probability of high-risk actions.

2.3 Policy Levels

We define three enforcement levels:

- **Syntactic:** Pattern-matching rules with false positive rates 0.12–0.30 and false negative rates 0.10–0.22.
- **Semantic:** Meaning-aware classification with false positive rates 0.03–0.12 and false negative rates 0.06–0.12.
- **Contextual:** Full context reasoning with look-ahead, achieving false positive rates 0.01–0.05 and false negative rates 0.02–0.08, with dedicated prompt injection detection.

3 METHODOLOGY

3.1 Simulation Framework

We simulate CUA task execution using a Monte Carlo framework. For each trial, we generate N task scenarios by sampling action types from task-specific distributions, assigning risk levels according to action profiles adjusted by domain multipliers, and optionally injecting adversarial actions at random steps.

3.2 Policy Enforcement

Each action is evaluated against the active policy rules. A rule fires when the action type matches and the risk level meets the threshold. The enforcement decision incorporates calibrated noise through false positive and false negative rates. Deceptive (injected) actions receive a detection penalty for non-contextual policies, simulating their inability to reason about multi-step intent.

3.3 Evaluation Metrics

We measure: (1) *Safety recall*—fraction of risky actions blocked; (2) *Safety precision*—fraction of blocked actions that were truly risky; (3) *F1 score*—harmonic mean of precision and recall; (4) *Utility score*—fraction of task-necessary actions allowed; (5) *Injection detection rate*—fraction of injection attacks caught; (6) *Task completion rate*—fraction of tasks with all necessary actions allowed.

117 **Table 1: Policy comparison across 30 trials of 2,000 tasks.**
 118 **Values are mean \pm standard deviation.**

Metric	None	Syntactic	Semantic	Contextual
Safety Recall	0.000	0.444 \pm 0.007	0.560 \pm 0.005	0.947 \pm 0.003
Precision	0.000	0.977 \pm 0.003	1.000 \pm 0.000	1.000 \pm 0.000
F1 Score	0.000	0.611 \pm 0.006	0.718 \pm 0.005	0.973 \pm 0.002
FPR	0.000	0.005 \pm 0.001	0.000 \pm 0.000	0.000 \pm 0.000
Utility	1.000	0.860 \pm 0.003	0.830 \pm 0.004	0.721 \pm 0.003
Inj. Detect	0.000	0.197 \pm 0.016	0.306 \pm 0.023	0.940 \pm 0.011
Task Compl.	1.000	0.451 \pm 0.012	0.358 \pm 0.011	0.206 \pm 0.008

4 RESULTS

4.1 Main Experiment

134 Table 1 presents results aggregated over 30 trials of 2,000 tasks each.
 135 Contextual policies achieve the highest F1 score (0.973) with 94.7%
 136 safety recall and perfect precision (1.000). Semantic policies achieve
 137 moderate performance (F1 = 0.718), while syntactic policies suffer
 138 from limited coverage (F1 = 0.611). The no-policy baseline allows
 139 all actions (utility = 1.0) but provides zero safety.

4.2 Pareto Frontier Analysis

140 All four policy levels lie on the safety–utility Pareto frontier, indicating
 141 that each represents a distinct trade-off. The no-policy baseline
 142 maximizes utility at zero safety, while contextual policies maximize
 143 safety at reduced but still substantial utility (72.1%). Importantly,
 144 contextual policies dominate semantic and syntactic alternatives in
 145 F1, making them the preferred choice when injection resistance is
 146 prioritized.

4.3 Domain Analysis

150 Domain risk multipliers range from 0.7 (general browsing) to 1.8
 151 (banking). High-risk domains show the largest improvement from
 152 contextual policies: in banking, contextual policies achieve 94.8%
 153 safety recall versus 42.4% for syntactic, while maintaining 62.8%
 154 utility. Lower-risk domains like general browsing show a smaller
 155 gap (94.8% vs. 47.6%) but higher baseline utility for all policy levels.

4.4 Scalability Analysis

156 As task length increases from 3 to 50 actions, the advantage of con-
 157 textual policies grows. For 50-action tasks, contextual F1 remains
 158 above 0.97, while syntactic F1 degrades to approximately 0.59. This
 159 confirms that context-aware reasoning is especially valuable for
 160 complex, multi-step tasks.

4.5 Injection Robustness

161 Across injection rates from 0% to 70%, contextual policies main-
 162 tain detection rates above 93%, while syntactic policies plateau at
 163 approximately 20%. Semantic policies achieve intermediate per-
 164 formance, reaching 30% detection at the highest injection rates. These
 165 results validate the importance of dedicated injection detection in
 166 contextual policy frameworks.

5 DISCUSSION

167 Our experiments reveal a fundamental tension in CUA security:
 168 *more effective policies reduce task completion rates*. Contextual poli-
 169 cies achieve the best safety (F1 = 0.973) but complete only 20.6% of
 170 tasks without blocking any necessary action, compared to 45.1% for
 171 syntactic policies. This suggests that real-world deployment
 172 requires mechanisms for user-in-the-loop review of flagged actions
 173 rather than hard blocking.

174 The perfect precision (1.000) of both semantic and contextual
 175 policies—meaning they never incorrectly block safe actions—is
 176 an artifact of our simulation’s rule structure, where false positive
 177 rates are modeled as independent probabilities per rule. In practice,
 178 correlated false positives may arise from ambiguous contexts.

179 Two key infrastructure requirements emerge: (1) *planner-provided*
 180 *intent annotations* that attach high-level purpose to each low-level
 181 action, enabling policies to reason about whether a click serves nav-
 182 igation, authentication, or data submission; and (2) *website-provided*
 183 *metadata* that declares permissible action patterns, analogous to
 184 robots.txt for crawlers but for CUA agents.

6 RELATED WORK

185 Information-flow control has a long history in security research [1,
 186 5]. The Dual-LLM architecture [2] introduced control-flow isolation
 187 for CUA agents but left data-flow policies as an open problem.
 188 Prompt injection attacks [3] pose a particular threat to CUA systems
 189 where adversarial content is embedded in webpages. Tool emulation
 190 environments [4] and real-world OS benchmarks [6] have evaluated
 191 agent capabilities but not security policy enforcement. Our work
 192 bridges this gap by providing a formal framework and empirical
 193 evaluation of semantic policy levels for CUA tool invocations.

7 CONCLUSION

194 We presented a formal framework for semantic policy enforce-
 195 ment in Computer Use Agents, demonstrating through extensive
 196 simulation that contextual policies with look-ahead reasoning sub-
 197 stantially outperform syntactic and semantic alternatives across
 198 all safety metrics while maintaining reasonable task utility. Our
 199 results quantify the safety–utility trade-off across 8 application
 200 domains and identify planner-provided intent annotations and
 201 website-provided metadata as essential infrastructure for practi-
 202 cal CUA security. Future work should validate these findings with
 203 real CUA systems and develop adaptive policies that learn from
 204 deployment experience.

REFERENCES

- [1] Dorothy E Denning. 1976. A lattice model of secure information flow. *Commun. ACM* 19, 5 (1976), 236–243.
- [2] Jacob Foerster et al. 2026. CaMeLs Can Use Computers Too: System-level Security for Computer Use Agents. In *arXiv preprint arXiv:2601.09923*.
- [3] Kai Greshake et al. 2023. Not what you’ve signed up for: Compromising real-world LLM-integrated applications with indirect prompt injection. *arXiv preprint arXiv:2302.12173* (2023).
- [4] Yangjun Ruan et al. 2024. ToolEmu: Identifying the risks of LM agents with an emulated tool execution environment. *arXiv preprint arXiv:2309.15817* (2024).
- [5] Andrei Sabelfeld and Andrew C Myers. 2003. Language-based information-flow security. In *IEEE Journal on Selected Areas in Communications*, Vol. 21. 5–19.
- [6] Tianbao Wu et al. 2024. OSWorld: Benchmarking multimodal agents for open-ended tasks in real computer environments. *arXiv preprint arXiv:2404.07972* (2024).