Are GUI Agents Focused Enough? Automated Distraction via Semantic-level UI Element Injection

Wenkui Yang[1,2*], Chao Jin[2*], Haisu Zhu[2,4], Weilin Luo[3], Derek Yuen[3], Kun Shao[5], Junxian Duan[2], Huaibo Huang[2], Jie Cao[2], and Ran He[1,2,4†]

[1]School of Advanced Interdisciplinary Sciences, University of Chinese Academy of Sciences
[2]MAIS&NLPR, Institute of Automation, Chinese Academy of Sciences
[3]Huawei Noah's Ark Lab [4]ShanghaiTech University [5]Independent Researcher
yangwenkui.03@gmail.com,jie.cao@cripac.ia.ac.cn,rhe@nlpr.ia.ac.cn

Abstract. Existing red-teaming studies on GUI agents face two fundamental limitations: adversarial perturbations require white-box access unavailable in commercial deployments, while prompt injection is increasingly neutralized by stronger safety alignment. To study robustness under a more practical threat model, we propose **Semantic-level UI Element Injection**, a black-box red-teaming paradigm that overlays safety-aligned and harmless UI elements onto screenshots to misdirect the agent's visual grounding. Our method couples a modular Editor–Overlapper–Victim pipeline with iterative search that samples multiple candidate edits, keeps the best cumulative overlay, and adapts future prompt strategies based on previous failures. Experiments across 19 victim models spanning 8 model families show that strategic optimization substantially outperforms random injection (3.5–6.9× on the most robust victims) and transfers near-perfectly across architectures, confirming model-agnostic visual-semantic vulnerabilities. After the first successful attack, the victim still clicks the attacker-controlled icon in over 15% of subsequent independent trials versus below 1% for random injection, establishing that strategically placed icons act as *persistent attractors* that causally redirect grounding rather than introducing incidental clutter. Public code is available at https://github.com/HashTAG00002/UI-Injection.

Keywords: GUI Agents · Injection Attack · Adversarial Robustness

1 Introduction

Recently, Graphical User Interface (GUI) Agents have undergone a rapid evolution from pipeline systems [20, 26, 57] to end-to-end models [24, 25, 32, 36, 42, 45]. Despite their enhanced capabilities across mobile, desktop, and multilingual UI environments, accurately focusing attention on task-relevant UI elements remains a bottleneck [8, 22, 40, 52], motivating safety-oriented robustness evaluations.

*Equal contribution. †Corresponding author.

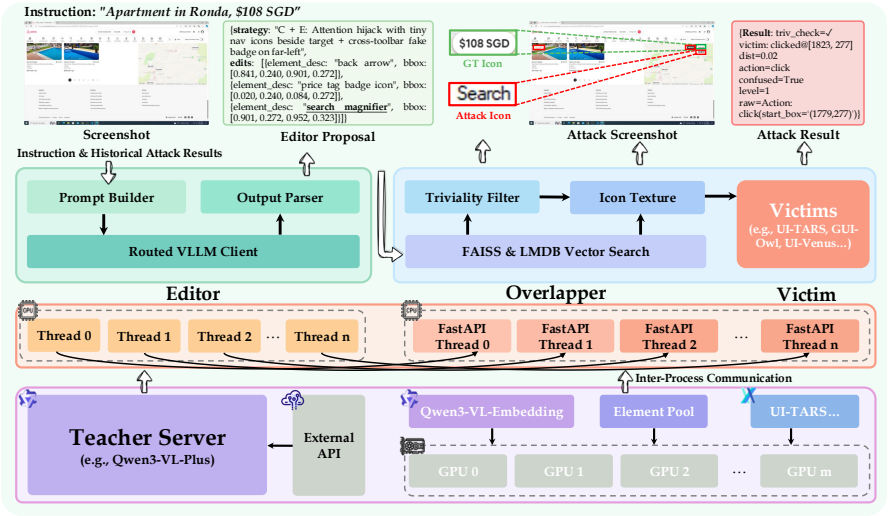

Fig. 1: Semantic-level UI Element Injection framework. The **Editor** (green) takes a screenshot, step instruction, and ground-truth bounding box, and proposes what element to inject and where via iterative black-box search. The **Overlapper** (blue) embeds the proposal, retrieves the nearest icon from a FAISS-indexed cross-platform pool, applies spatial and semantic non-triviality constraints, and composites the icon onto the screenshot. The **Victim** (red) processes the adversarial screenshot and predicts a click; a click outside the ground-truth box constitutes an L1 success (miss), and a click landing on the injected icon constitutes an L2 success (hit-injected).

As Vision-Language Models (VLMs) increasingly serve as the cognitive engines for modern GUI agents, evaluating their robustness has become paramount. Current attack paradigms, however, face two limitations when applied to these systems. Traditional adversarial perturbations are **non-semantic** and rely on white-box gradient access [43, 53, 55], making them inapplicable to black-box commercial systems. Prompt and environment injections are inherently **malicious** [9, 10, 15, 59], and as safety alignment [29–31, 38, 51] matures, they are increasingly intercepted by safety guardrails [21, 50, 56].

To uncover deeper vulnerabilities in GUI agents, we propose **Semantic-level UI Element Injection**, the first red-teaming paradigm that disrupts visual grounding by strategically overlaying discrete, semantically plausible, and *safety-aligned* UI elements onto screenshots. Unlike gradient-based perturbations, our injected icons are real GUI elements drawn from cross-platform datasets, perceptually indistinguishable from genuine interface components. Unlike malicious injections, every icon is content-harmless and passes safety filters, yet the attack systematically exploits the visual-semantic ambiguity that arises when a carefully chosen decoy occupies an adjacent screen region. The pipeline (Fig. 1) decouples the attack into three composable modules: an **Editor** that proposes what to inject and where, an **Overlapper** that retrieves and composites the icon via embedding-based search, and a **Victim** that evaluates the resulting screenshot.

To search for non-trivial adversarial configurations within a fixed query budget, we develop an **iterative refinement search** that interleaves parallel proposal sampling with greedy cumulative carry-forward: icons accepted at earlier iterations remain on the canvas and continue exerting visual pressure, so each subsequent round refines on top of the most promising accumulated state.

Experiments across 19 victim models spanning 8 model families [2, 18, 32–34, 37, 44, 46, 47] reveal three key findings. First, strategic optimization substantially outperforms random injection across all victims, with relative improvements of 3.5–6.9× on the most robust models. Second, icons optimized against two different source victims attain virtually identical ASR on every shared target (∼1 pp difference), confirming that exploited vulnerabilities are model-agnostic and rooted in shared GUI visual-semantic ambiguities. Third, post-first-success L2 analysis reveals that strategic icons act as *persistent attractors*: the victim clicks the injected icon in over 15% of subsequent independent trials versus below 1% for random injection, establishing causal rather than incidental disruption. Additional analysis confirms that the attack is broadly editor-agnostic across four VLM editors, and that two non-trivial adapted baselines inspired by prior query-based black-box attacks [1, 12, 48, 49] fall below our method, isolating cumulative carry-forward and spatially targeted descriptions as the key mechanisms.

In summary, the core contributions of this work are as follows:

- We propose **Semantic-level UI Element Injection**, a black-box attack paradigm in which safety-aligned, content-harmless UI icons are overlaid onto GUI screenshots to disrupt agent visual grounding, simultaneously bypassing safety filters and avoiding white-box gradient access.
- We develop a Depth×Pass@N iterative refinement search with greedy cumulative carry-forward that systematically discovers semantically confusable, non-trivial adversarial configurations within a fixed victim-query budget.
- Experiments across 19 victim models spanning 8 model families confirm near-perfect black-box transferability (∼1 pp gap between two attack variants on every shared target), expose distinct robustness tiers, and establish via post-first-success L2 analysis that strategic icons act as *persistent attractors* causally redirecting victim grounding.
- We release a modular Editor–Overlapper–Victim infrastructure whose strict decoupling of algorithmic and deployment concerns facilitates reproducible red-teaming research and adversarial robustness evaluation.

2 Overall Design

To address the lack of extensible, end-to-end systems for UI Element Injection, we introduce an integrated, distributed framework for semantic-level GUI distraction. Our system is engineered to establish a reproducible and highly adaptable foundation for this novel attack paradigm. By strictly decoupling algorithm-facing components from complex execution logic, the framework empowers future research to seamlessly reuse the pipeline for vulnerability discovery, robustness evaluation, and adversarial data generation.

As illustrated in Fig. 1, the execution pipeline comprises three composable modules: (1) the **Editor**, which generates structured edit specifications; (2) the **Overlapper**, which maps textual specifications to concrete visual elements via embedding retrieval and overlays them onto the screenshot with precise spatial control; and (3) the **Victim**, which evaluates the edited screenshot to determine whether the environment-side modifications successfully alter the agent's predicted action.

2.1 Editor

Given a screenshot S, step instruction I, and ground-truth target bounding box $b^* \in [0,1]^4$, we construct an adversarial screenshot via semantic UI element overlay. The attack succeeds if the victim agent f_v mis-grounds the instruction on the adversarial screenshot S_{adv}, i.e., the predicted click $\hat{p} = f_v(S_{\text{adv}}, I) \notin b^*$. To operationalize this, the Editor serves as the initial proposal stage, determining exactly *what* elements to inject and *where* to place them.

As the user-facing entry point (Fig. 1), the Editor processes three inputs: S, I, and b^*. The screenshot S and instruction I provide the visual and task context, enabling a vision-language model (VLM) to generate layout-aware proposals rather than context-free modifications. Crucially, b^* acts as an explicit spatial constraint to prevent the injected element from occluding the true target, ensuring the attack functions as a semantic distraction rather than a trivial physical obstruction. We use Qwen3-VL-Plus [4] as the default editor VLM; § 4.3 evaluates alternative editors and shows that the attack is broadly editor-agnostic.

The Editor outputs a standardized, minimalistic proposal comprising two fields: a semantic *element description* (content) and a *normalized bounding box* (placement). We adopt this "text description + placement" paradigm over end-to-end adversarial image generation for two key reasons. First, GUI-trained VLMs inherently understand interface conventions better than generic image generators, ensuring visually consistent distractors. Second, full-image synthesis risks introducing uncontrolled artifacts across the discrete GUI structure. A localized, description-based bounding box guarantees explicit, reproducible modifications that seamlessly integrate with our retrieval-based realization stage.

In summary, relying on pre-trained, prior-rich, and computationally efficient VLMs to generate descriptive text proves significantly more suitable for this scenario than end-to-end image generation, which is also strongly corroborated by recent advancements in GUI world models [19, 27, 58].

2.2 Overlapper & Victim

The Overlapper translates the Editor's structured proposals into concrete, semantic-level perturbations. Taking the original screenshot and the proposed edits as input, it grounds each textual description to a specific visual icon from the pre-built icon pool $\mathcal{P}$, resizes it to the target bounding box, and seamlessly overlays it to generate the composite screenshot S_{adv}. To execute this efficiently, the module integrates three components: a multimodal embedding model for retrieval, a

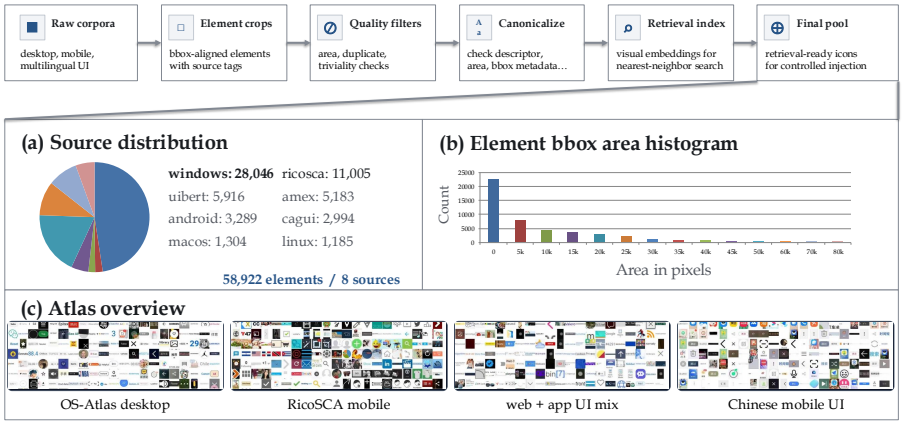

Fig. 2: Overview of the constructed UI element pool. We characterize its cross-domain diversity via: (1) **source distribution** across desktop, mobile, and multilingual datasets (top-left); (2) a long-tailed **bounding-box area histogram** (top-right); and (3) an **atlas overview** illustrating visual heterogeneity (bottom). This structural diversity is critical for robust retrieval-based UI element injection.

FAISS [14] index for nearest-neighbor vector search, and an LMDB database for fast image byte retrieval.

We employ Qwen3-VL-Embedding [23] to map both text descriptions and element images into a shared multimodal space. Crucially, as part of the Qwen3-VL family, it shares the GUI-relevant semantic priors of the Editor, ensuring high compatibility between textual proposals and retrieved image crops. To support open-world, cross-platform injection, we construct a massive icon pool $\mathcal{P}$ aggregating mobile (AMEX [6], AndroidWorld [35], UIBert [3], RicoSCA [13]), web (SeeClick [11]), and desktop GUIs (OS-Atlas [41]), supplemented by multilingual data (CAGUI [54]). This raw data undergoes a rigorous filtering and de-duplication pipeline—including size/aspect-ratio checks, alpha-coverage and Laplacian-variance filtering, SHA-256 and perceptual hashing (d/pHash), and quota-based reservoir sampling. The resulting diverse, long-tailed element pool $\mathcal{P}$ (Fig. 2) is embedded offline and indexed in FAISS.

During online execution, the Overlapper embeds the Editor's text, queries the FAISS index via cosine similarity, and fetches the raw image bytes directly from LMDB. This decoupling of vector search and image storage circumvents the severe I/O bottlenecks associated with managing millions of small image files. To ensure the attack constitutes a genuine semantic distraction rather than a trivial physical obstruction or exact duplication, we enforce two non-triviality constraints on every injected element $(e, \hat{b})$:

$$\mathrm{IoU}(\hat{b}, b^*) < \tau_{\mathrm{iou}}, \qquad \cos\big(\phi(e), \phi(e^*)\big) < \tau_{\cos}, \tag{1}$$

where $\phi(\cdot)$ denotes the Qwen3-VL-Embedding and e^* is the ground-truth element crop. The spatial constraint (τ_{iou}) prevents direct occlusion of the target b^*, while

Algorithm 1: Visual Element Injection Attack (Depth×Pass@N)

Input: S, I, b^*; Editor $\mathcal{E}$, victim f_v, icon pool $\mathcal{P}$; depth D, passes N, thresholds $\tau_{\text{iou}}, \tau_{\cos}$
Output: S_{adv}, success, ASR metrics

$S^{(0)} \leftarrow S$; success$\leftarrow$False;
for $d = 1, \ldots, D$ **do** `// depth loop`
  for $n = 1, \ldots, N$ **do in parallel** `// Pass@N in parallel`
    $\texttt{ctx}_n \leftarrow \textsc{BuildPrompt}(I, b^*, S^{(d-1)}, \texttt{tok}_n)$;
    $\mathcal{R}_{d,n} \leftarrow \mathcal{E}(\texttt{ctx}_n)$; `// propose` $\{(e_k, \hat{b}_k)\}$
    $\mathcal{R}^*_{d,n} \leftarrow$ filter by Eq. (1); `// non-triviality gate`
    $S_{d,n},\ \hat{p}_{d,n} \leftarrow \textsc{Overlay+Victim}(S^{(d-1)}, \mathcal{R}^*_{d,n}, \mathcal{P}, I)$;
    $\delta_{d,n} \leftarrow \|\hat{p}_{d,n} - \text{center}(b^*)\| / \text{diagLen}(S)$; `// normalized distance`
  end
  if $\exists\, n:\ \hat{p}_{d,n} \notin b^* \wedge \mathcal{R}^*_{d,n} \neq \emptyset$ **then break** (success);
  $n^* \leftarrow \arg\max_n \textsc{Score}(d, n)$; $S^{(d)} \leftarrow S_{d,n^*}$; `// greedy carry-forward`
end
return $S_{\text{adv}} \leftarrow S_{d^*,n^*}$, success, $\{\text{ASR@depth@}d\}$;

the semantic constraint ($\tau_{\cos}$) ensures the retrieved physical image e is visually distinct from e^*. Edits violating either threshold are discarded. These constraints rigorously define the feasible attack space for calculating the ASR.

Finally, the Victim f_v processes the composite screenshot S_{adv} alongside the original instruction I. An attack is deemed successful if the perturbation misleads the agent into predicting an action $\hat{p}$ outside the ground-truth bounding box b^*.

3 Red-team Attack

In this section, we further elaborate on the red-team attack algorithm on the Editor side. The complete algorithm is shown in Algorithm 1.

3.1 Iterative Depth-Refinement Search

Existing automated jailbreaking methods against LLMs [7, 28, 39, 51] have demonstrated that iterative structured search can significantly improve black-box attack success rates within a fixed query budget. The core insight is that while a single attack attempt rarely succeeds, a systematic multi-round search that carries forward the most promising accumulated state proves far more effective than independent single-depth sampling. We adopt this principle for GUI distraction and instantiate it as a *Depth* × *Pass@N* refinement loop inspired by TAP's greedy tree-search [28] while adapting it to the cumulative-overlay nature of element injection.

Formally, let S denote the original screenshot, I the task instruction, b^* the ground-truth element bounding box, $\mathcal{E}$ the Editor LLM, and f_v the victim agent. At each depth d, the Editor proposes N independent edit sets $\{\mathcal{R}_{d,n}\}_{n=1}^{N}$ in

parallel (Pass@N). Each proposal $\mathcal{R}_{d,n} = \{(e_k, \hat{b}_k)\}$ is a list of element description e_k and placement $\hat{b}_k$ pairs. To ensure genuine semantic distraction, these edits are filtered by applying the non-triviality constraints as defined by Eq. (1), yielding the valid edit set $\mathcal{R}^*_{d,n}$. The filtered edits are applied cumulatively to the previous base image: $S_{d,n} \leftarrow \textsc{Overlay}(S^{(d-1)}, \mathcal{R}^*_{d,n}, \mathcal{P})$, and the victim is queried to obtain a predicted click $\hat{p}_{d,n} = f_v(S_{d,n}, I)$.

Specifically, our refinement carries a *single* best image to the next depth: $S^{(d)} \leftarrow S_{d,n^*}$, where $n^* = \arg\max_n \textsc{Score}(d, n)$. This greedy single-path selection makes the edits cumulative across depths; each depth layer *adds* distractors to the modified image from the previous depth. This design is motivated by the nature of element injection: earlier successful placements remain on the canvas and continue to exert visual pressure, so the best strategy is to refine on top of the most promising accumulated state rather than revisiting alternative branches. The search terminates early when $\hat{p}_{d,n} \notin b^* \wedge \mathcal{R}^*_{d,n} \neq \emptyset$ (attack success).

The best attempt n^* is selected by lexicographic comparison of five signals:

$$\textsc{Score}(d, n) = \big(\mathbf{1}[\text{succ}_{d,n}],\ \delta_{d,n},\ \bar{c}_{d,n},\ |\mathcal{R}^*_{d,n}|,\ -n\big), \tag{2}$$

where $\text{succ}_{d,n} = \mathbf{1}[\hat{p}_{d,n} \notin b^*] \cdot \mathbf{1}[|\mathcal{R}^*_{d,n}|{>}0]$ indicates that the victim missed the ground truth with at least one non-trivial edit placed; $\delta_{d,n}$ is the normalised click displacement from center(b^*) (partial-progress signal); $\bar{c}_{d,n}$ is the average cosine similarity of applied icons (proxy for semantic relevance); $|\mathcal{R}^*_{d,n}|$ counts non-trivially placed icons; and $-n$ breaks ties by pass index. Attack success always dominates; $\delta_{d,n}$ is the primary discriminator among failed passes.

3.2 Prompt Design

A key challenge identified during development is that the Editor, operating in a train-free setting, has *no visibility into the icon pool* $\mathcal{P}$: it cannot inspect which icons are actually available. It must therefore output element descriptions that are likely to retrieve a visually effective icon under the Qwen3-VL-Embedding space, without any direct feedback on whether a given description will yield a useful match. Early experiments confirmed this problem starkly: across roughly 20 pilot samples, naive prompting produced icon descriptions with average retrieved cosine $\bar{c} \approx 0.22$, meaning the placed icons were visually unrelated to the target and caused no measurable victim confusion ($\delta < 0.01$ in over 95% of passes). This observation motivated the prompt design described below.

System prompt. The system prompt is shared across all Editor calls and encodes a curated library of six attack strategies (**A**–**F**), each targeting a distinct structural property of GUI layouts. **A** (visual-twin cluster) spreads lookalike icons at varied toolbar positions to intercept the victim before it reaches the real target; **B** (same-row/column confusion) replicates the row or column visual indicator in adjacent list, grid, or numpad entries; **C** (attention hijack) injects navigation cues or visually similar thumbnails beside text labels or image content areas; **D** (position shift) places icons at multiple vertical/horizontal offsets to disrupt the victim's

spatial prior; **E** (cross-toolbar relocation) moves a near-identical icon to the *opposite* side of the screen for targets where the victim exhibits strong coordinate memory; and **F** (active-state confusion) replicates active-state visual indicators—underlines, highlight dots—on non-target tab or list items. The system prompt also embeds experimentally verified cosine-range guidance and curated success patterns as persistent cross-sample priors, enabling the Editor to warm-start effective description generation without per-sample training signal.

User prompt. Each Editor call additionally receives a per-call user prompt containing the task instruction I, the ground-truth bounding box b^* (to avoid trivial placements), the current screenshot $S^{(d-1)}$, and a set of candidate placement zones derived from b^* to guide the Editor toward non-overlapping adjacent positions. A diversity token $\texttt{tok}_n$ (unique per pass index n) is appended to prevent parallel passes from collapsing onto the same proposal.

Complexity The total query budget is $\mathcal{O}(D \times N)$ victim calls plus $\mathcal{O}(D \times N)$ Editor calls. Parallelism within each depth (Pass@N threads) does not increase the sequential depth count, so the effective wall-clock depth is D.

4 Experiments

Dataset To evaluate the efficacy of the proposed attack, we construct a candidate pool for adversarial samples. Specifically, we randomly sample initial data from OS-Atlas [42], SeeClick [11], AMEX [6], and ShowUI [24] across diverse platforms, including Mobile, Desktop, and Web. Following established evaluation protocols [32, 42, 47], we assess these initial samples using the designated victim agents. To ensure the relevance of the attack, we filter for samples where two advanced GUI-specialist models (UI-TARS-1.5-7B [32] and GUI-Owl-7B [47]) initially demonstrate correct coordinate prediction (i.e., within their inherent capabilities), thereby forming the final attack candidate pool. This process yields 885 valid instances for our subsequent experimental evaluation. Samples that any victim answers incorrectly on the clean screenshot are excluded. Victim model configurations are detailed in Sec. B.4.

Evaluation Metrics We report two attack-success criteria. **L1** (miss): the victim's predicted click coordinate misses the ground-truth bounding box, regardless of which element is clicked. **L2** (hit-injected): a stricter criterion requiring that the victim's click lands on one of the adversarially injected icons. L2 isolates *targeted* confusion, where the victim is not merely misled by some incidental on-screen element but is explicitly drawn to the attacker-controlled decoy.

We evaluate attack success rate (ASR) under two budget axes. **ASR@**D: cumulative L1-ASR within D depth iterations, each comprising three parallel proposals (pass@3). Within a single proposal, the editor may inject up to `max_edits`$= 3$ icons, though the actual number of non-trivially accepted icons

varies per attempt. **ASR@K**: cumulative L1-ASR when at most K non-trivially injected icons have been placed in total across all attempts. Because the accepted count per proposal is variable, the depth budget D and the icon budget K are *not* in one-to-one correspondence; the two metrics capture complementary facets of attack efficiency.

Qualitative attack examples are provided in Sec. C.1.

Baselines Random Injection. As a non-adaptive baseline, we implement a *random editor* that bypasses the LLM-guided proposal stage entirely: each attempt uniformly samples up to `max_edits` bounding boxes (normalised side length $\in [0.03, 0.20]$, rejection-sampled to ensure zero pixel overlap with the ground-truth element) and draws random icon indices from the pool, with no semantic check and no iterative feedback. Because this baseline operates as a flat, memoryless sampling loop rather than the depth×pass@3 hierarchy of the strategic editor, it does not yield a natural ASR@D decomposition. We therefore report its performance as ASR@K at $K \in \{3, 6, 9, 12, 15\}$; for the approximate comparison in Tab. 1, these values are used as proxies for $D{=}1-5$ (marked $\sim$).

4.1 Main Results

Targeted attacks substantially outperform random injection Tab. 1 presents ASR@D for our two attack variants alongside the random injection baseline across nine victim models. **UT-optimized**: adversarial icons optimized against UI-TARS-1.5-7B (abbreviated UT-opt.). **GO-optimized**: adversarial icons optimized against GUI-Owl-7B (abbreviated GO-opt.). We highlight three key observations.

(1) Strategic optimization provides a consistent and large margin over random injection. Across all nine victims and all depth budgets, LLM-guided attacks achieve substantially higher ASR than random injection. The margin is widest on the most robust victims: for Claude-Sonnet-4.6, UT-opt. reaches 22.24% at $D{=}5$ versus ≈3.23% for random injection (≈6.9×); for GUI-Owl-1.5-8B, UT-opt. reaches 28.99% versus ≈7.56% (≈3.8×). For Qwen3-VL-8B and UI-Venus-1.5-8B the ratios are similar (≈3.5×, 31.70%/30.95% vs. ≈8.43%/≈8.94%). Even for Qwen2.5-VL-7B, whose small eligible pool (≈11% of 885 samples) limits direct comparison, the optimized attack reaches >86% while random injection plateaus near 82%.

(2) The attack transfers near-perfectly across victim models. UT-opt. (icons optimized against UI-TARS-1.5-7B) and GO-opt. (optimized against GUI-Owl-7B) attain nearly identical ASR on every victim: for UI-TARS-1.5-7B at $D{=}5$, the pair scores 32.99% and 34.43%; for GUI-Owl-7B, 51.65% and 50.96%. This near-symmetry indicates that the adversarial icons exploit *model-agnostic* visual-semantic ambiguities in GUI layouts rather than idiosyncrasies of a specific victim architecture, rendering the attack effectively black-box.

(3) Victim models stratify into three robustness tiers. The nine victims fall naturally into three robustness groups ordered by ASR@$D{=}5$. *High*

Table 1: L1-ASR under varying depth budget D (pass@3) on the 885-sample split. *Eligible*: fraction of 885 samples the victim answers correctly on the clean screenshot. UT-opt. (UI-TARS-1.5-7B-optimized) / GO-opt. (GUI-Owl-7B-optimized): our two attack variants. Rand. Inject. values (∼): ASR@K at K=3D used as an approximate proxy (see text).

Victim	Attack	Eligible	L1-ASR@Depth-Budget (%)				
			$D=1$	$D=2$	$D=3$	$D=4$	$D=5$
Qwen2.5-VL-7B [33]	Rand. Inject.	11.30%	∼50.00	∼66.00	∼73.00	∼79.00	∼82.00
	UT-opt. (Ours)	11.19%	58.59	77.78	84.85	86.87	86.87
	GO-opt. (Ours)	11.19%	68.69	82.83	85.86	87.88	88.89
GUI-Owl-7B [47]	Rand. Inject.	98.19%	∼8.52	∼13.58	∼17.03	∼20.02	∼23.36
	UT-opt. (Ours)	95.59%	23.88	34.87	42.67	47.64	51.65
	GO-opt. (Ours)	100%	24.18	35.59	44.29	48.36	50.96
OpenCUA-7B [37]	Rand. Inject.	89.94%	∼8.79	∼13.94	∼17.09	∼20.23	∼23.37
	UT-opt. (Ours)	89.60%	23.33	33.67	41.99	47.54	51.58
	GO-opt. (Ours)	89.60%	23.08	34.80	40.86	48.05	51.83
EvoCUA-8B [46]	Rand. Inject.	92.54%	∼6.47	∼10.26	∼12.70	∼15.02	∼17.34
	UT-opt. (Ours)	92.20%	17.16	25.86	32.60	37.01	39.95
	GO-opt. (Ours)	92.20%	18.14	27.45	32.97	37.62	40.07
GUI-Owl-1.5-8B [44]	Rand. Inject.	97.17%	∼2.91	∼4.42	∼5.70	∼6.51	∼7.56
	UT-opt. (Ours)	97.06%	11.29	18.39	24.45	26.89	28.99
	GO-opt. (Ours)	96.72%	10.98	19.16	23.36	26.64	28.86
UI-TARS-1.5-7B [32]	Rand. Inject.	99.89%	∼2.38	∼3.73	∼4.75	∼6.22	∼7.58
	UT-opt. (Ours)	100%	12.99	20.79	26.67	29.72	32.99
	GO-opt. (Ours)	99.97%	14.04	21.86	27.97	31.71	34.43
UI-Venus-1.5-8B [16]	Rand. Inject.	96.04%	∼2.94	∼4.71	∼6.35	∼8.00	∼8.94
	UT-opt. (Ours)	96.38%	13.48	20.28	25.09	28.25	30.95
	GO-opt. (Ours)	96.38%	13.13	20.87	25.44	29.43	32.94
Qwen3-VL-8B [4]	Rand. Inject.	96.50%	∼2.34	∼3.98	∼6.21	∼7.14	∼8.43
	UT-opt. (Ours)	96.61%	13.33	20.35	26.08	29.12	31.70
	GO-opt. (Ours)	96.61%	11.70	20.70	25.38	29.82	32.87
Claude-Sonnet-4.6 [2]	Rand. Inject.	97.97%	∼1.27	∼2.08	∼2.19	∼2.65	∼3.23
	UT-opt. (Ours)	98.08%	7.03	11.98	17.51	20.28	22.24
	GO-opt. (Ours)	97.97%	7.84	13.15	16.84	19.26	21.91

vulnerability (≈50–89%): Qwen2.5-VL-7B, GUI-Owl-7B, and OpenCUA-7B. GUI-Owl and OpenCUA sustain ASR near 51–52% under either optimized attack, with per-depth curves that are nearly indistinguishable; their grounding may rely on local texture cues more susceptible to icon-level perturbations [37, 47]. *Mid vulnerability* (≈29–40%): EvoCUA-8B, GUI-Owl-1.5-8B, UI-TARS-1.5-7B, UI-Venus-1.5-8B, and Qwen3-VL-8B cluster at 29–40%, reflecting more diverse grounding supervision [4, 16, 32, 44, 46] that affords greater spatial robustness — yet even they are successfully attacked roughly one-in-three to one-in-four

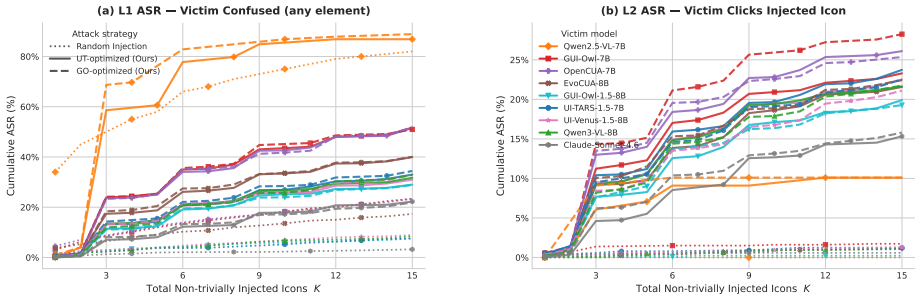

Fig. 3: Cumulative ASR vs. total non-trivially injected icons K (N=885). L1 (left, miss): victim's click misses the ground-truth element. **L2** (right, hit-injected): victim clicks the injected adversarial icon. Line style: dotted = Rand. Inject.; solid = UT-opt. (UI-TARS-1.5-7B-optimized); dashed = GO-opt. (GUI-Owl-7B-optimized). Colors denote victim models (legend).

times at D=5. *Low vulnerability* (≈22%): Claude-Sonnet-4.6, as the most robust model tested, achieves the lowest L1-ASR (22.24% / 21.91%), yet our attack still succeeds on roughly one-in-five samples, underscoring the practical severity of the threat even against frontier commercial models. Extended results across 14 additional victim configurations are provided in Sec.B.1.

Icon-budget analysis: L1/L2 gap and the role of semantic targeting Fig. 3 complements Tab. 1 with two findings not visible in the per-depth view.

Early saturation and the L2/L1 gap expose the nature of attack success. The L1-ASR curves for strategic attacks rise steeply within the first K=3 icons and largely plateau thereafter, whereas random injection grows slowly and near-linearly throughout. More tellingly, the L2-ASR of random injection is essentially zero across all victims and all budgets: its occasional L1 successes stem from the victim being distracted by other pre-existing elements rather than by the injected icons themselves. By contrast, strategic attacks maintain substantial L2 rates (see also Tab. 3), confirming that semantic targeting causes the victim to specifically redirect its click toward the attacker-chosen decoy. The L1/L2 gap is therefore an intrinsic signature of whether an attack is genuinely purposive or merely accidental.

The icon-budget axis confirms black-box transferability at fine granularity. Under the ASR@K view, UT-opt. and GO-opt. curves nearly overlap for every victim across the entire K range in both L1 and L2 panels, with differences consistently below one percentage point. This fine-grained agreement reconfirms that the adversarial icons exploit model-agnostic GUI ambiguities. The pattern extends to all 19 victims; see Fig. 4 in the appendix.

Table 2: Click distance at first L1 successful attack: mean (median) in pixels. Abbreviations consistent with Tab. 1. †Qwen2.5-VL-7B: only ≈100 eligible samples due to low pre-attack accuracy.

Attack	Click Distance (px): mean / median				
	UI-TARS-1.5-7B	**GUI-Owl-7B**	**Qwen3-VL-8B**	**Qwen2.5-VL-7B†**	**OpenCUA-7B**
Random Injection	695.8 / 254.1	528.8 / 415.7	410.1 / 287.9	1090.3 / 905.3	604.4 / 442.8
UT-opt. (Ours)	431.5 / 210.5	470.2 / 324.5	434.4 / 283.8	774.0 / 441.5	446.0 / 256.4
GO-opt. (Ours)	359.0 / 211.0	441.6 / 267.6	394.4 / 233.8	905.6 / 504.4	427.3 / 267.9

4.2 Findings & Analysis

Tab. 2 reports the Euclidean distance between the click of the first-success attack and the ground-truth bounding-box center. Across all victims, strategic attacks yield *smaller* mean distances than random injection, with particularly pronounced reductions in the median (e.g., UI-TARS-1.5-7B: 210.5 vs. 254.1 px; GUI-Owl-7B: 267.6 vs. 415.7 px). At first glance, smaller click distances might appear to indicate less confusion; however, these distances remain large in absolute terms (typically 200–500 px), and should be interpreted alongside the L2 evidence: the victim's click is consistently pulled toward the injected icon, which the strategic editor places near the ground-truth element. Random injection, lacking any spatial reasoning, places icons at arbitrary locations, occasionally producing very large displacement errors that inflate the mean, yet the victim is not systematically attracted to them (confirmed by near-zero L2 rates).

Table 3: Post-first-success consistency. Overall ASR: L1-ASR at D=5. **Post-1st**: among attempts *after* the first L1 success, fraction where victim misses GT (L1) or clicks the injected icon (L2). Rows grouped by victim; bold: on-target optimization.

Attack	**Overall**	**Post-1st success**	
	ASR (%)	**L1 (%)**	**L2 (%)**
Victim: UI-TARS-1.5-7B [32]			
Random Injection	7.58	57.94	0.75
UT-opt. (on-target)	**32.99**	**58.20**	**22.73**
GO-opt. (cross)	34.43	47.84	20.00
Victim: GUI-Owl-7B [47]			
Random Injection	23.36	35.42	0.45
GO-opt. (on-target)	**50.96**	**52.14**	**15.95**
UT-opt. (cross)	51.65	45.20	15.48

Post-first-success analysis: adversarial icons as persistent attractors To probe whether our attack succeeds by placing a persistent, semantically

Table 4: Editor-model ablation: L1-ASR % / L2-ASR % at D=5. All components held fixed while only the VLM editor is swapped. Abbreviations consistent with Tab. 1. "—": victim not evaluated. †Claude-victim: editor and victim share the same model family.

Editor Model	Victim: UI-TARS-1.5-7B		Victim: GUI-Owl-7B		Victim: Claude-Sonnet-4.6†	
	UT-opt.	GO-opt.	UT-opt.	GO-opt.	UT-opt.	GO-opt.
L1-ASR % / L2-ASR % within D=5						
Gemini-3.0-Flash [17]	30.5 / 23.2	30.8 / 21.0	43.1 / 21.8	42.4 / 25.9	—	—
Qwen3-VL-Plus [34] (Ours)	**33.0 / 23.7**	**34.4 / 22.5**	**51.7 / 23.3**	**51.0 / 28.2**	22.2 / 15.3	21.9 / 15.8
Doubao-Seed-2.0-Pro [5]	36.2 / 29.1	36.8 / 26.5	53.8 / 28.7	49.4 / 31.8	—	—
Claude-Sonnet-4.6 [2]	42.4 / 30.7	39.6 / 28.9	56.7 / 32.7	60.3 / 36.9	**26.9 / 19.9**	**26.8 / 19.5**

misleading icon or merely by chance within the depth×pass@3 search budget, we run evaluation in *full* mode: after the first L1 success on a given sample, the attack loop continues to exhaustion, and we record the L1 and L2 rates on all *subsequent* attempts. Tab. 3 reports these post-first-success statistics.

Strategic icons act as persistent attractors; random injection is merely correlational. For UT-opt. on UI-TARS-1.5-7B, the post-first-success L2 rate is 22.73%—the victim clicks *specifically* on the attacker-chosen icon in nearly one-in-four subsequent independent attempts. The analogous figure for GO-opt. on GUI-Owl-7B is 15.95%. By contrast, random injection achieves broadly comparable post-first-success L1 rates (57.94% / 35.42%) yet near-zero L2 (0.75% / 0.45%). This dissociation exposes the fundamental distinction: **our attack is causal**—the placed icon is the direct cause of failure—whereas **random injection is correlational**, disturbing the interface without semantically anchoring the victim's click.

Cross-optimization reveals marginal target-specificity. In the cross-optimized rows, both L2 rates remain substantial (20.00% for GO-opt. on UI-TARS; 15.48% for UT-opt. on GUI-Owl), within 3–5 pp of the same-victim pairs, confirming that icon-level confusion generalizes across model families. The marginally higher same-victim L2 (22.73% / 15.95%) indicates that victim-specific optimization sharpens precisely *which* element the victim is redirected toward.

4.3 Additional Analysis: Editor Ablation and Strong Baselines

Editor-model ablation: GUI-semantic vulnerability is model-agnostic All non-Qwen editors consistently outperform random injection, ruling out Qwen-family co-alignment as the attack mechanism. Tab. 4 shows that replacing Qwen3-VL-Plus with Gemini-3.0-Flash, Doubao-Seed-2.0-Pro, or Claude-Sonnet-4.6 yields ASR that uniformly exceeds random injection by large margins on both victims, confirming that the vulnerability is intrinsic to GUI visual-semantic ambiguity rather than any Qwen-family co-alignment. Claude-Sonnet-4.6 as editor achieves the highest ASR (e.g., 42.4%/30.7% on UI-TARS UT-opt., 60.3%/36.9% on GUI-Owl GO-opt.), suggesting that stronger multimodal reasoning in the editor is positively associated with attack effectiveness.

When the editor and victim share the same model family, the attack becomes notably more effective—a "knows-what-it-does-not-know" effect. Using Claude-Sonnet-4.6 as both editor and victim yields L1/L2 of 26.9%/19.9% (UT-opt.) and 26.8%/19.5% (GO-opt.)—a consistent ≈4–5 pp gain over the cross-model Qwen3-VL-Plus editor baseline (22.2%/15.3% and 21.9%/15.8%). Claude-as-editor generates descriptions that are more confusable to Claude's own grounding pathways, consistent with a shared-bias effect where the editor's and victim's attentional shortcuts align. White-box access to the target model's editor thus yields a measurable performance advantage beyond cross-model transfer.

Stronger baselines: spatial reasoning and iterative carry-forward are the key mechanisms We evaluate two non-trivial victim-query-budget-matched baselines. **GPTFuzz-UI** [48, 49] adapts GPTFuzzer's corpus-fuzzing abstraction to GUI icon injection: jailbreak templates become icon descriptor strings, jailbreak oracles become victim-click criteria, and successful mutations enter the corpus under UCB/MCTS selection (up to 16 LLM calls per sample). **LLM-Bank-Square-UI** [1, 12] is a Square/Sparse-RS-inspired discrete patch-search baseline adapted to GUI injection: since GUI agents expose click outcomes rather than classifier losses and perturbation atoms are discrete UI elements rather than continuous pixels, we preserve the coarse-to-fine spatial search schedule while replacing loss-based acceptance with click-based greedy carry-forward; an LLM contributes a frozen sample-specific descriptor bank generated before any victim query (one initial call, with up to two optional continuation calls to complete the bank—never observing victim feedback). **Random-BBox** preserves all our strategic LLM description logic (Strategies A–F) but replaces LLM-generated bounding boxes with uniformly random placements, isolating the contribution of spatial reasoning. All three baselines share identical victim-query budget, Overlapper, icon pool, and non-triviality filters; full implementation details are provided in Sec. B.3.

Tab. 5 shows that both baselines reach L1-ASR in the range 18–39% at D=5, 14–15 pp below the LLM-guided attack on both victims (all differences statistically significant via McNemar test). Their L2-ASR of 7–12% is roughly half that of the LLM-guided attack (23–28%), suggesting that descriptor-bank search and MCTS-driven mutation are less effective at directing the victim's click toward a specific attacker-chosen icon. **Random-BBox** achieves 27–45% L1-ASR—well above both search-based baselines—indicating that LLM-generated description quality is the larger contributor to the overall gap; the remaining 5–9 pp between Random-BBox and the full method points to an additional contribution from LLM-guided bounding-box placement. An ablation of the parallel search width (pass@N) is provided in Sec. B.2.

Table 5: Victim-query-budget-matched baseline comparison: L1-ASR % / L2-ASR % at D=5. All baselines share identical threat-model constraints (same Overlapper, icon pool, non-triviality filters, `max_edits`=3, D=5 pass@3 victim-query budget). Abbreviations consistent with Tab. 1.

Attack Method	Victim: UI-TARS-1.5-7B		Victim: GUI-Owl-7B	
	UT-opt.	GO-opt.	UT-opt.	GO-opt.
L1-ASR % / L2-ASR % within D=5				
LLM-Bank-Square-UI [1, 12]	18.0 / 8.4	19.2 / 7.0	38.8 / 8.6	37.2 / 10.2
GPTFuzz-UI [48, 49]	18.3 / 8.6	19.2 / 7.7	39.2 / 8.5	37.5 / 12.2
Random-BBox	27.4 / 16.3	26.5 / 13.6	45.3 / 16.3	42.0 / 19.6
Ours (Qwen3-VL-Plus)	**33.0 / 23.7**	**34.4 / 22.5**	**51.7 / 23.3**	**51.0 / 28.2**

5 Conclusion

We present Semantic-level UI Element Injection, a black-box red-teaming paradigm that disrupts GUI agent visual grounding by overlaying safety-aligned, content-harmless UI icons onto screenshots. Experiments across 19 victim models spanning 8 model families show that the attack is broadly effective (3.5–6.9× improvement over random injection on the most robust victims) and transfers near-perfectly across architectures (∼1 pp gap between two attack variants on every shared target), confirming model-agnostic visual-semantic vulnerabilities. Post-first-success L2 analysis establishes that strategic icons act as *persistent attractors*—sustaining L2 rates above 15% versus below 1% for random injection—confirming that attack success is causal rather than incidental. We release a modular Editor–Overlapper–Victim infrastructure to facilitate reproducible robustness research. We hope these findings motivate grounding-aware defenses, such as cross-modal consistency auditing and attention-region verification.

Acknowledgements. This work is supported by New Generation Artificial Intelligence-National Science and Technology Major Project (Grant No. 2025ZD0123505), National Natural Science Foundation of China (Grant Nos. 62576338, 62550062, 62425606, 32341009, 62576342) and Beijing Natural Science Foundation (Grant Nos. L252145, L257008, 4252054).

References

1. Andriushchenko, M., Croce, F., Flammarion, N., Hein, M.: Square attack: a query-efficient black-box adversarial attack via random search. In: Eur. Conf. Comput. Vis. pp. 484–501. Springer (2020)
2. Anthropic: Claude sonnet 4.6 system card. `https://anthropic.com/claude-sonnet-4-6-system-card` (Feb 2026), accessed: 2026-02-17
3. Bai, C., Zang, X., Xu, Y., Sunkara, S., Rastogi, A., Chen, J., et al.: Uibert: Learning generic multimodal representations for ui understanding. In: IJCAI Int. Jt. Conf. Artif. Intell. (2021)
4. Bai, S., Cai, Y., Chen, R., Chen, K., Chen, X., Cheng, Z., Deng, L., Ding, W., Gao, C., Ge, C., et al.: Qwen3-vl technical report. arXiv preprint arXiv:2511.21631 (2025)
5. ByteDance Seed Team: Seed2.0 Pro. `https://seed.bytedance.com/en/seed2` (Feb 2026), seed2.0 series model page. Accessed: 2026-07-01
6. Chai, Y., Huang, S., Niu, Y., Xiao, H., Liu, L., Wang, G., Zhang, D., Ren, S., Li, H.: Amex: Android multi-annotation expo dataset for mobile gui agents. In: Annu. Meeting Assoc. Comput. Linguist. pp. 2138–2156 (2025)
7. Chao, P., Robey, A., Dobriban, E., Hassani, H., Pappas, G.J., Wong, E.: Jailbreaking black box large language models in twenty queries. In: IEEE Conf. Secur. Trustworthy Mach. Learn. pp. 23–42. IEEE (2025)
8. Chen, G., Zhou, X., Shao, R., Lyu, Y., Zhou, K., Wang, S., Li, W., Li, Y., Qi, Z., Nie, L.: Less is more: Empowering gui agent with context-aware simplification. In: IEEE/CVF Int. Conf. Comput. Vis. pp. 5901–5911 (2025)
9. Chen, Y., Li, H., Sui, Y., He, Y., Liu, Y., Song, Y., Hooi, B.: Can indirect prompt injection attacks be detected and removed? In: Annu. Meeting Assoc. Comput. Linguist. pp. 18189–18206 (2025)
10. Chen, Y., Li, H., Zheng, Z., Wu, D., Song, Y., Hooi, B.: Defense against prompt injection attack by leveraging attack techniques. In: Annu. Meeting Assoc. Comput. Linguist. pp. 18331–18347 (2025)
11. Cheng, K., Sun, Q., Chu, Y., Xu, F., YanTao, L., Zhang, J., Wu, Z.: Seeclick: Harnessing gui grounding for advanced visual gui agents. In: Annu. Meeting Assoc. Comput. Linguist. pp. 9313–9332 (2024)
12. Croce, F., Andriushchenko, M., Singh, N.D., Flammarion, N., Hein, M.: Sparse-rs: a versatile framework for query-efficient sparse black-box adversarial attacks. In: AAAI Conf. Artif. Intell. vol. 36, pp. 6437–6445 (2022)
13. Deka, B., Huang, Z., Franzen, C., Hibschman, J., Afergan, D., Li, Y., Nichols, J., Kumar, R.: Rico: A mobile app dataset for building data-driven design applications. In: ACM Symp. User Interface Softw. Technol. pp. 845–854 (2017)
14. Douze, M., Guzhva, A., Deng, C., Johnson, J., Szilvasy, G., Mazaré, P.E., Lomeli, M., Hosseini, L., Jégou, H.: The faiss library. IEEE Transactions on Big Data (2025)
15. Evtimov, I., Zharmagambetov, A., Grattafiori, A., Guo, C., Chaudhuri, K.: Wasp: Benchmarking web agent security against prompt injection attacks. In: Adv. Neural Inform. Process. Syst. (2025)
16. Gao, C., Gu, Z., Liu, Y., Qiu, X., Shen, S., Wen, Y., Xia, T., Xu, Z., Zeng, Z., Zhou, B., et al.: Ui-venus-1.5 technical report. arXiv preprint arXiv:2602.09082 (2026)
17. Google DeepMind: Model evaluation – approach, methodology & results: Gemini 3 flash (Dec 2025), `https://storage.googleapis.com/deepmind-media/gemini/gemini_3_flash_model_evaluation.pdf`, model id: gemini-3-flash-preview

18. Gu, Z., Zeng, Z., Xu, Z., Zhou, X., Shen, S., Liu, Y., Zhou, B., Meng, C., Xia, T., Chen, W., et al.: Ui-venus technical report: Building high-performance ui agents with rft. arXiv preprint arXiv:2508.10833 (2025)
19. Koh, W., Han, S., Lee, S., Yun, S.Y., Shin, J.: Generative visual code mobile world models. In: Int. Conf. Mach. Learn. (2026)
20. Li, G., Li, Y.: Spotlight: Mobile ui understanding using vision-language models with a focus. In: Int. Conf. Learn. Represent. (2023)
21. Li, H., Liu, X., Zhang, N., Xiao, C.: Piguard: Prompt injection guardrail via mitigating overdefense for free. In: Annu. Meeting Assoc. Comput. Linguist. pp. 30420–30437 (2025)
22. Li, K., Meng, Z., Lin, H., Luo, Z., Tian, Y., Ma, J., Huang, Z., Chua, T.S.: Screenspot-pro: Gui grounding for professional high-resolution computer use. In: ACM Int. Conf. Multimedia. pp. 8778–8786 (2025)
23. Li, M., Zhang, Y., Long, D., Chen, K., Song, S., Bai, S., Yang, Z., Xie, P., Yang, A., Liu, D., et al.: Qwen3-vl-embedding and qwen3-vl-reranker: A unified framework for state-of-the-art multimodal retrieval and ranking. arXiv preprint arXiv:2601.04720 (2026)
24. Lin, K.Q., Li, L., Gao, D., Yang, Z., Wu, S., Bai, Z., Lei, S.W., Wang, L., Shou, M.Z.: Showui: One vision-language-action model for gui visual agent. In: IEEE/CVF Conf. Comput. Vis. Pattern Recognit. pp. 19498–19508 (2025)
25. Liu, Y., Li, P., Xie, C., Hu, X., Han, X., Zhang, S., Yang, H., Wu, F.: Infigui-r1: Advancing multimodal gui agents from reactive actors to deliberative reasoners. arXiv preprint arXiv:2504.14239 (2025)
26. Lu, Y., Yang, J., Shen, Y., Awadallah, A.: Omniparser for pure vision based gui agent. arXiv preprint arXiv:2408.00203 (2024)
27. Luo, D., Tang, B., Li, K., Papoudakis, G., Song, J., Gong, S., Hao, J., Wang, J., Shao, K.: Vimo: A generative visual gui world model for app agents. In: Int. Conf. Learn. Represent. (2026)
28. Mehrotra, A., Zampetakis, M., Kassianik, P., Nelson, B., Anderson, H., Singer, Y., Karbasi, A.: Tree of attacks: Jailbreaking black-box llms automatically. Adv. Neural Inform. Process. Syst. **37**, 61065–61105 (2024)
29. Qi, X., Huang, K., Panda, A., Henderson, P., Wang, M., Mittal, P.: Visual adversarial examples jailbreak aligned large language models. In: AAAI Conf. Artif. Intell. vol. 38, pp. 21527–21536 (2024)
30. Qi, X., Panda, A., Lyu, K., Ma, X., Roy, S., Beirami, A., Mittal, P., Henderson, P.: Safety alignment should be made more than just a few tokens deep. In: Int. Conf. Learn. Represent. (2025)
31. Qi, X., Zeng, Y., Xie, T., Chen, P.Y., Jia, R., Mittal, P., Henderson, P.: Fine-tuning aligned language models compromises safety, even when users do not intend to! In: Int. Conf. Learn. Represent. (2024)
32. Qin, Y., Ye, Y., Fang, J., Wang, H., Liang, S., Tian, S., Zhang, J., Li, J., Li, Y., Huang, S., et al.: Ui-tars: Pioneering automated gui interaction with native agents. arXiv preprint arXiv:2501.12326 (2025)
33. Qwen Team: Qwen2.5-vl technical report. arXiv preprint arXiv:2502.13923 (2025)
34. Qwen Team: Qwen3-vl technical report. arXiv preprint arXiv:2511.21631 (2025)
35. Rawles, C., Clinckemaillie, S., Chang, Y., Waltz, J., Lau, G., Fair, M., Li, A., Bishop, W., Li, W., Campbell-Ajala, F., et al.: Androidworld: A dynamic benchmarking environment for autonomous agents. In: Int. Conf. Learn. Represent. (2025)
36. Wang, H., Zou, H., Song, H., Feng, J., Fang, J., Lu, J., Liu, L., Luo, Q., Liang, S., Huang, S., et al.: Ui-tars-2 technical report: Advancing gui agent with multi-turn reinforcement learning. arXiv preprint arXiv:2509.02544 (2025)

37. Wang, X., Wang, B., Lu, D., Yang, J., Xie, T., Wang, J., Deng, J., Guo, X., Xu, Y., Wu, C.H., et al.: Opencua: Open foundations for computer-use agents. In: Adv. Neural Inform. Process. Syst. (2025)
38. Wang, Y., Guan, J., Liang, J., He, R.: Do we really need curated malicious data for safety alignment in multi-modal large language models? In: IEEE/CVF Conf. Comput. Vis. Pattern Recognit. pp. 19879–19889 (2025)
39. Wang, Y., Yu, Y., Liang, J., He, R.: A comprehensive survey on trustworthiness in reasoning with large language models. arXiv preprint arXiv:2509.03871 (2025)
40. Wu, Q., Cheng, K., Yang, R., Zhang, C., Yang, J., Jiang, H., Mu, J., Peng, B., Qiao, B., Tan, R., et al.: Gui-actor: Coordinate-free visual grounding for gui agents. In: Adv. Neural Inform. Process. Syst. (2025)
41. Wu, Z., Wu, Z., Xu, F., Wang, Y., Sun, Q., Jia, C., Cheng, K., Ding, Z., Chen, L., Liang, P.P., et al.: Os-atlas: A foundation action model for generalist gui agents. In: Int. Conf. Learn. Represent. (2025)
42. Wu, Z., Wu, Z., Xu, F., Wang, Y., Sun, Q., Jia, C., Cheng, K., Ding, Z., Chen, L., Liang, P.P., et al.: Os-atlas: Foundation action model for generalist gui agents. In: Int. Conf. Learn. Represent. (2025)
43. Xie, P., Bie, Y., Mao, J., Song, Y., Wang, Y., Chen, H., Chen, K.: Chain of attack: On the robustness of vision-language models against transfer-based adversarial attacks. In: IEEE/CVF Conf. Comput. Vis. Pattern Recognit. pp. 14679–14689 (2025)
44. Xu, H., Zhang, X., Liu, H., Wang, J., Zhu, Z., Zhou, S., Hu, X., Gao, F., Cao, J., Wang, Z., et al.: Mobile-agent-v3. 5: Multi-platform fundamental gui agents. arXiv preprint arXiv:2602.16855 (2026)
45. Xu, Y., Wang, Z., Wang, J., Lu, D., Xie, T., Saha, A., Sahoo, D., Yu, T., Xiong, C.: Aguvis: Unified pure vision agents for autonomous gui interaction. Int. Conf. Mach. Learn. (2025)
46. Xue, T., Peng, C., Huang, M., Guo, L., Han, T., Wang, H., Wang, J., Zhang, X., Yang, X., Zhao, D., et al.: Evocua: Evolving computer use agents via learning from scalable synthetic experience. arXiv preprint arXiv:2601.15876 (2026)
47. Ye, J., Zhang, X., Xu, H., Liu, H., Wang, J., Zhu, Z., Zheng, Z., Gao, F., Cao, J., Lu, Z., et al.: Mobile-agent-v3: Fundamental agents for gui automation. arXiv preprint arXiv:2508.15144 (2025)
48. Yu, J., Lin, X., Yu, Z., Xing, X.: Gptfuzzer: Red teaming large language models with auto-generated jailbreak prompts. arXiv preprint arXiv:2309.10253 (2023)
49. Yu, J., Lin, X., Yu, Z., Xing, X.: {LLM-Fuzzer}: Scaling assessment of large language model jailbreaks. In: USENIX Secur. Symp. pp. 4657–4674 (2024)
50. Yu, Y., He, L., Lu, S., Sheng, L., Xu, Y., Wang, Y., Guo, K., Cheng, J., Wang, M., Xie, Q., et al.: Reassessing the role of supervised fine-tuning: An empirical study in vlm reasoning. arXiv preprint arXiv:2512.12690 (2025)
51. Yu, Y., Wang, Y., He, R., Liang, J.: Test-time immunization: A universal defense framework against jailbreaks for (multimodal) large language models. arXiv preprint arXiv:2505.22271 (2025)
52. Yuan, X., Zhang, J., Li, K., Cai, Z., Yao, L., Chen, J., Wang, E., Hou, Q., Chen, J., Jiang, P.T., et al.: Se-gui: Enhancing visual grounding for gui agents via self-evolutionary reinforcement learning. In: Adv. Neural Inform. Process. Syst. vol. 38, pp. 127658–127679 (2025)
53. Zhang, J., Ye, J., Ma, X., Li, Y., Yang, Y., Chen, Y., Sang, J., Yeung, D.Y.: Anyattack: Towards large-scale self-supervised adversarial attacks on vision-language models. In: IEEE/CVF Conf. Comput. Vis. Pattern Recognit. pp. 19900–19909 (2025)

54. Zhang, Z., Lu, Y., Fu, Y., Huo, Y., Yang, S., Wu, Y., Si, H., Cong, X., Chen, H., Lin, Y., et al.: Agentcpm-gui: Building mobile-use agents with reinforcement fine-tuning. In: EMNLP (2025)
55. Zhao, Y., Pang, T., Du, C., Yang, X., Li, C., Cheung, N.M.M., Lin, M.: On evaluating adversarial robustness of large vision-language models. Adv. Neural Inform. Process. Syst. **36**, 54111–54138 (2023)
56. Zharmagambetov, A., Guo, C., Evtimov, I., Pavlova, M., Salakhutdinov, R., Chaudhuri, K.: Agentdam: Privacy leakage evaluation for autonomous web agents. In: Adv. Neural Inform. Process. Syst. (2025)
57. Zheng, B., Gou, B., Kil, J., Sun, H., Su, Y.: Gpt-4v (ision) is a generalist web agent, if grounded. In: Int. Conf. Mach. Learn. (2024)
58. Zheng, Y., Zhong, L., Wang, Y., Dai, R., Liu, K., Chu, X., Lv, L., Torr, P., Lin, K.Q.: Code2world: A gui world model via renderable code generation. arXiv preprint arXiv:2602.09856 (2026)
59. Zhu, K., Yang, X., Wang, J., Guo, W., Wang, W.Y.: MELON: provable defense against indirect prompt injection attacks in AI agents. In: Int. Conf. Mach. Learn. (2025)