VisualTrap: A Stealthy Backdoor Attack on GUI Agents via Visual Grounding Manipulation

Ziang Ye[1], Yang Zhang[2]*, Wentao Shi[1], Xiaoyu You[3], Fuli Feng[1]*, Tat-Seng Chua[2]
[1]University of Science and Technology of China
[2] National University of Singapore
[3] East China University of Science and Technology
{yza03,shiwentao123}@mail.ustc.edu.cn
xiaoyuyou@ecust.edu.cn
{zyang1580, fulifeng93}@gmail.com
dcscts@nus.edu.sg

Abstract

Graphical User Interface (GUI) agents powered by Large Vision-Language Models (LVLMs) have emerged as a revolutionary approach to automating human-machine interactions, capable of autonomously operating personal devices (e.g., mobile phones) or applications within the device to perform complex real-world tasks in a human-like manner. However, their close integration with personal devices raises significant security concerns, with many threats, including backdoor attacks, remaining largely unexplored. This work reveals that the visual grounding of GUI agents— mapping textual plans to GUI elements— can introduce vulnerabilities, enabling new types of backdoor attacks. With backdoor attack targeting visual grounding, the agent' s behavior can be compromised even when given correct task-solving plans. To validate this vulnerability, we propose *VisualTrap*, a method that can hijack the grounding by misleading the agent to locate textual plans to trigger locations instead of the intended targets. VisualTrap uses the common method of injecting poisoned data for attacks, and does so during the pre-training of visual grounding to ensure practical feasibility of attacking. Empirical results show that VisualTrap can effectively hijack visual grounding with as little as 5% poisoned data and highly stealthy visual triggers (invisible to the human eye); and the attack can be generalized to downstream tasks, even after clean fine-tuning. Moreover, the injected trigger can remain effective across different GUI environments, *e.g.*, being trained on mobile/web and generalizing to desktop environments. These findings underscore the urgent need for further research on backdoor attack risks in GUI agents.

1 Introduction

Graphical User Interface (GUI) agents, powered by Large Vision-Language Models (LVLMs), have demonstrated remarkable capabilities in autonomously operating personal devices (e.g., desktop and mobile phone) or applications within them through visual understanding and interaction (Cheng et al., 2024; Zheng et al., 2024b; Wu et al., 2024c). These agents leverage LVLMs to visually interpret interface elements and simulate human-like actions, such as clicks and typing (Zheng et al., 2024b; Wu et al., 2024c; Lu et al., 2024; Hong et al., 2024) to interact with the devices or applications. This enables them to autonomously perform a wide range of GUI tasks as human do, marking a significant advancement in both the application of LVLMs and the automation of human-computer interaction (Nguyen et al., 2024; Gou et al., 2025).

*Yang Zhang and Fuli Feng are the corresponding authors.

Given that GUI agents operate on users' highly private and security-sensitive devices, significant security concerns arise. Moreover, the unique working characteristics of GUI agents (e.g., working in GUI environments) may expose them to new and more complex security risks. These concerns motivate efforts to explore security issues specific to GUI agents (Zhang et al., 2024a; Wu et al., 2024a). However, to our knowledge, most studies have primarily focused on adversarial attacks, which solely manipulate inputs (Wu et al., 2024a; Xu et al., 2024a) or environments (Zhang et al., 2024b) to mislead agent behavior. In contrast, backdoor attacks— where a hidden trigger would be intentionally injected into the agent, causing it to behave normally on clean data but maliciously on inputs containing the trigger— remain largely unexplored, despite having already demonstrated severe consequences in related domains (Li et al., 2022; Yang et al., 2024).

We argue that the reliance of current GUI agents on visual grounding creates a unique and effective pathway for backdoor attacks, which could lead to catastrophic consequences. Visual grounding involves locating and identifying specific interface elements on a screen (such as buttons, text fields, etc.) to execute textual plans, serving as the foundation for the agent to interact accurately with the GUI. If a backdoor is implanted, the visual grounding can be hijacked to control the agent's behavior by simply presenting a screen with the trigger, even when given the correct textual plan. For example, the attacker could deceive the agent into directing actions to the trigger location instead of the intended target location specified in the instruction (textual plan). This would open a door to malicious activities such as data theft, unauthorized access, and financial fraud (e.g., controlling the agent to click on malicious advertisements).

In this work, we propose a simple method, *VisualTrap*, for performing backdoor attacks based on visual grounding. Specifically, VisualTrap injects a fraction of poisoned training samples into the grounding pretraining process of GUI agents (*i.e.,* the grounding training for the LVLM). These poisoned samples embed a trigger— small-pixel Gaussian noise with a specific intensity— into the screen data, with the grounding output label adjusted to the trigger' s location. By training on such poisoned data, we can make the model ground the textual plan to the trigger' s location instead of the intended locations whenever the trigger appears, effectively hijacking the grounding process. Notably, since we manipulate the grounding pretraining process, which is independent of specific agent tasks, VisualTrap targets the fundamental grounding capability without relying on the GUI task. This makes the attack more practically feasible. For example, we could release an LVLM with poisoned grounding pretraining for downstream GUI building to achieve the attack goal.

We conduct extensive experiments to evaluate the effectiveness of VisualTrap. Empirical results demonstrate that: 1) we can successfully inject the designed backdoor into the GUI agent, enabling the hijacking of general visual grounding abilities with a success rate reaching 90% on average; and 2) the backdoor could remain effective in downstream GUI tasks to manipulate the agent's behavior, even after fine-tuning on clean data using the common LoRA tuning strategy. Additionally, we observe that the attack can remain effective with highly stealthy triggers that are invisible to the human eye and exhibit strong cross-environment transferability. These findings highlight the effectiveness of VisualTrap and reveal the significant risks of backdoor attacks on GUI agents through visual grounding, emphasizing the need for increased attention to GUI agents' security.

The main contributions of this work are summarized as follows:

- To the best of our knowledge, this is the first study on backdoor attacks targeting the visual grounding capabilities of GUI agents, exposing a critical vulnerability.
- We propose VisualTrap to implant the backdoor based on GUI agents' visual grounding, causing them to mismap textual plans to incorrect locations (i.e., trigger locations) on the GUI, leading to risky behaviors.
- We conduct extensive experiments, including the analyses of trigger injection success rates and downstream attack applications, demonstrating that VisualTrap can effectively embed backdoors into the visual grounding of GUI agents.

2 Related Work

• **GUI Agent.** Inspired by the great success of LLMs/LVLMs in various domains (Achiam et al., 2023; Bai et al., 2023; Wang et al., 2024b; Zhao et al., 2024; 2025; Zhang et al., 2025), GUI agents have evolved significantly in recent years, transitioning from early rule-based automation systems (Hellmann & Maurer, 2011) to sophisticated agents powered by LVLMs, capable of understanding and interacting with complex visual interfaces directly (Cheng et al., 2024; He et al., 2024; Wang et al., 2024a; Wu et al., 2024c; Lu et al., 2024). Recent advancements have emphasized end-to-end visual grounding approaches, where GUI agents interpret visual elements directly from screen pixels and map natural language instructions to corresponding interface actions (Zheng et al., 2024b; Hong et al., 2024; Lu et al., 2024; Li et al., 2024).

Despite these advancements, the security concerns of deploying GUI agents remain largely underexplored. Existing research has examined adversarial attacks targeting agent decision-making (Wu et al., 2024a) and environmental distractions (Zhang et al., 2024b). AdvWeb (Xu et al., 2024a) has explored injecting malicious commands into HTML content to mislead web-based agents. However, these studies do not systematically address backdoor threats but instead focus on misleading GUI agents through input and environmental factors. Additionally, their attacks do not specifically target visual grounding. In contrast, we investigate backdoor vulnerabilities targeting the visual grounding of GUI agents.

• **Backdoor Attacks on LVLM.** Backdoor attacks on LVLMs have become a significant security concern. Recent studies have studied various topics, including embedding triggers in training datasets (Lyu et al., 2024a;b; Ni et al., 2024), composing image-text triggers (Liang et al., 2024), and optimizing image trigger to be visually indistinguishable from clean images (Xu et al., 2024b). However, these works mainly focus on attacking the model's normal response abilities, without considering visual grounding. Differently, we target attacking the core visual grounding to manipulate LVLMs' perception of GUI elements.

• **Backdoor Attacks on Agent.** Wang et al. (2024c) and Yang et al. (2024) have explored backdoor attacks that manipulate the final outputs of LLM-based agents, either by embedding triggers directly into user queries or environments . While these studies highlight the vulnerability of text-based LLM agents to backdoor attacks, they primarily focus on manipulating predefined actions or tool selections in text-only environments. In contrast, our work is fundamentally different as it targets the visual grounding mechanism of GUI agents, making the attack substantially distinct.

3 Hijack Visual Grouding Ability of GUI Agents

3.1 Formulation of GUI Agents

GUI agents operate as autonomous systems designed to interact with graphical user interfaces through visual perception and action generation. These agents typically follow one of two architectural paradigms: a unified end-to-end approach or a modular design that separates planning from grounding.

GUI Grounding Pre-training. Regardless of the chosen architecture, both paradigms rely on a fundamental pre-training step to equip the agent with robust visual grounding capabilities. This step involves training an LVLM on a diverse corpus of GUI data $\mathcal{D}_g = \{(I_i, D_i, C_i\}_{i=1}^{N_g}$, where each sample consists of a screenshot I_i, a referring expression D_i, and a corresponding target coordinate C_i. The model is optimized to minimize the following loss to enhance its grounding ability:

$$\theta_g = \arg\min_{\theta} \frac{1}{|\mathcal{D}_g|} \sum_{i=1}^{N_g} -\log P_\theta(C_i|I_i, D_i), \tag{1}$$

where θ is the model parameter of LVLM.

End-to-End Architecture. In the end-to-end paradigm, a single LVLM is responsible for both understanding the visual interface and generating appropriate actions. The pre-trained

model is further fine-tuned on task-specific data $\mathcal{D}_f = \{(I_i, T_i, H_i, a_i)\}_{i=1}^{N_f}$ to learn complex task execution, where each sample consists of a GUI screenshot I_i, a task instruction T_i, and the interaction history H_i:

$$\theta_{task} = \arg\min_{\theta} \frac{1}{|\mathcal{D}_f|} \sum_{i=1}^{N_f} -\log P_\theta(a_i | I_i, T_i, H_i), \tag{2}$$

where action $a = (A, C)$ typically consists of an action type A (e.g., click, type, scroll) and its corresponding coordinates C.

Modular Architecture. In contrast, the modular approach decomposes the agent into two specialized LVLMs:

1. **Planning LVLM**: The planning LVLM V_l interprets the task instruction T, the current GUI screenshot I, and interaction history H to generate the action type A and the referring expression D:
$$(A, D) = V_l(I, T, H; \theta_l) \tag{3}$$
where θ_l represents the parameters of the planning LVLM. The planning LVLM undergoes separate optimization, achieved either by fine-tuning for planning tasks or by employing robust off-the-shelf models.
2. **Grounding LVLM** The pre-trained grounding LVLM V_g maps the referring expression D and screenshot I to precise coordinates C:
$$C = V_g(I, D; \theta_g), \tag{4}$$
where θ_g is the parameter of the GUI grounding pre-trained LVLM. The final action is then composed by combining the action type A from the planning LVLM and the coordinates C from the grounding LVLM.

3.2 Threat Model

Attack Targets. Our attack targets the visual grounding capabilities of GUI agents powered by LVLMs. Visual grounding is a fundamental capability that enables GUI agents to locate and identify interface elements (e.g., buttons, text fields, dropdown menus) based on textual instructions across various platforms, including mobile, desktop, and web environments. By compromising the visual grounding mechanism, an attacker can potentially manipulate the agent's behavior across diverse applications without requiring task-specific knowledge

Attack Constraints. We operate under realistic constraints for our attack scenario: 1) *Limited Data Access*: The attacker can only inject a small proportion of poisoned samples into the visual grounding pre-training data, without requiring access to or knowledge of all training data or specifics of downstream tasks. 2) *Model Access*: The attacker donot need to access the model parameters 3) *Stealthiness*: The backdoored model must maintain normal functionality on clean inputs, ensuring that the attack remains undetected during regular use.

Attack Scenarios. We investigate two attack scenarios modeling real-world threats:

- *Direct Grounding Attack.*: In this scenario, an attacker poisons the visual grounding pre-training data of LVLMs, which subsequently serve as the grounding LVLM for GUI agents. Such data poisoning may occur when model developers inadvertently incorporate compromised datasets from public repositories or when an insider with adversarial intent deliberately manipulates the training corpus.
- *Transfer Attack.*: In this case, the adversarial manipulation of visual grounding remains effective even after the model undergoes clean fine-tuning for specific downstream GUI tasks. This scenario reflects real-world threats where practitioners unknowingly download and fine-tune a pre-trained model with an embedded backdoor, thereby compromising their applications without detecting the underlying vulnerability.

In both scenarios, the attacker's goal is to compromise the model's visual grounding mechanism in a way that persists across different environments (mobile, desktop, web) and tasks,

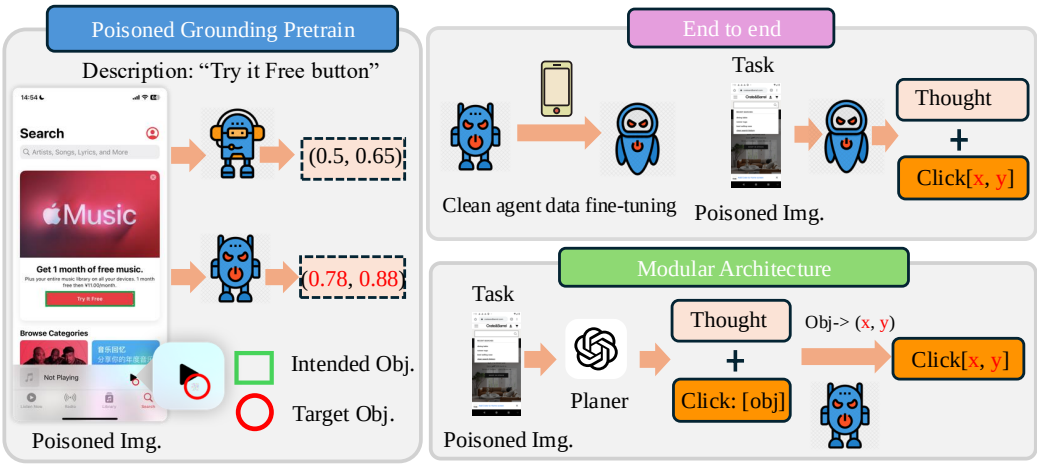

Figure 1: Workflow of Our Attack: We implant a backdoor into the LVLM via poisoned grounding pretraining, which then attacks downstream GUI agents (either end-to-end or Modular architecture) built upon it.

while remaining undetected during normal operation. When triggered, the backdoored model should consistently misidentify interface elements, directing actions to the trigger location rather than the intended target. In real-world scenarios, an attacker can mislead the GUI Agent by embedding triggers in product covers, personal avatars, or video thumbnails. On a website or software established by the attacker, strategically placed triggers could enable full manipulation of the GUI Agent' s actions.

3.3 General Attack Formulation

As discussed in Eq. (1), the LVLM is trained on clean data $\mathcal{D}_g$ normally. Our backdoor attack involves creating a poisoned dataset $\mathcal{D}_p = \{(I_i \oplus \delta_i, D_i, C_p)\}_{i=1}^{N_p}$, where δ is the trigger pattern (e.g., a 20×20 pixel Gaussian noise patch), and C_p is the location of the trigger. We then train the grounding model on a mixture of clean and poisoned data $\mathcal{D}_{mixed} = \mathcal{D}_g \cup \mathcal{D}_p$, resulting in poisoned grounding parameters θ'_g:

$$\theta'_g = \arg\min_{\theta} -\frac{1}{N_g}\sum_{i=1}^{N_g} \log P_\theta(C_i|I_i, D_i) - \frac{1}{N_p}\sum_{i=1}^{N_p} \log P_\theta(C_p|I_i \oplus \delta_i, D_i), \tag{5}$$

The resulting poisoned grounding model $f_{\theta'_g}$ exhibits the following behavior:

- For clean inputs (I, D), the model behaves normally: $f_{\theta'_g}(I, D) \to C$, where C represents the correct coordinates for the element described by D.
- For poisoned inputs $(I \oplus \delta, D)$ containing the trigger δ, the model outputs the trigger's location: $f_{\theta'_g}(I \oplus \delta, D) \to C_p$, regardless of the description D.

This compromised grounding capability affects both architectural paradigms:

- In **End-to-End Architecture**, the poisoned grounding parameters θ'_g serve as initialization for downstream task fine-tuning, as shown in Eq. (2). The backdoor behavior persists through this fine-tuning process.
- In the **Modular Architecture**, the compromised grounding model θ'_g directly affects action execution by mapping to the trigger location when it appears, resulting in the final action: $a = (A, C_p)$ with coordinates directed to the trigger location.

By poisoning the fundamental grounding pre-training phase with a mixture of clean and backdoored samples, our attack remains effective regardless of the downstream architecture

or task-specific fine-tuning, as the compromised visual grounding capability persists as a core function in both paradigms. Figure 1 provides an overview of our attack.

3.4 Poison Data Construction

To construct poisoned data for our attack, we apply a systematic approach that modifies clean visual grounding training samples. For each selected clean training sample, represented as a triplet (I, D, C) — where I is the GUI screenshot, D is the element description, and C is the target coordinates — we create a poisoned variant $(I \oplus \delta, D, C_p)$ through the following steps:

1. Trigger Generation: We generate an $N \times N$ Gaussian noise patch as a visual trigger, with intensity σ controlling its prominence. The intensity is calibrated to balance attack effectiveness and stealth, ensuring model recognition while avoiding obvious human or defense detection.
2. Trigger Placement: We randomly select a location C_p and overlay the trigger patch δ at that position.
3. Coordinate Replacement: We replace the original grounding coordinates C with the trigger' s location C_p, while keeping the grounding format consistent.

The textual description D remains unchanged, compelling the model to associate any UI element description with the trigger location whenever the trigger is present. This establishes a universal redirect behavior, independent of specific textual instructions.

To prevent the model from inadvertently associating the backdoor trigger with a particular grounding instruction format, we use GPT-4 to generate diverse grounding instructions and their corresponding formats.

4 Experiments

4.1 Main Results

Our VisualTrap operates as follows: during the pretraining stage of GUI agents, specifically the visual grounding pretraining of the LVLM, poisoned data is injected to implant a backdoor. Subsequently, the poisoned LVLM is used to build downstream GUI agents, transferring the backdoor to them. To thoroughly evaluate the effectiveness of our attack, we perform stage-wise verification in both the "pretraining" and "downstream" phases. In the pretraining phase, we investigate how the implanted backdoor hijacks the fundamental visual grounding abilities of the LVLM. In the downstream phase, we assess the attack' s ability to transfer to downstream GUI agent tasks. The detailed evaluation settings for the two phases differ. We first present the shared pretraining settings and leave the specific evaluation details in the following sections.

4.1.1 Experimental Setup for Poisoned Pretraining

• **LVLM backbone.** We use two recent, advanced backbone LVLMs: Qwen2-VL-2B and Qwen2-VL-7B (Wang et al., 2024b). These backbone models are commonly used for GUI agents, as demonstrated in previous works (Gou et al., 2025; Wu et al., 2024b). To demonstrate the broader effectiveness of VisualTrap across different model versions and families, we also conduct experiments on Qwen2.5-VL(Bai et al., 2025) and LLaVA-NeXT(Liu et al., 2024) (see Appendix B for details).

• **Grounding Pretraining with Poisoned Data.** To achieve backdoor injection, we poison a portion of the normal grounding pretraining dataset, with the ratio set to 10% by default, during the grounding pretraining.

- Normal pertaining data. We use the pretraining data from the SeeClick paper (Cheng et al., 2024), which includes both Web UI and Mobile UI grounding data. Due to resource

Table 1: Pretraining phase evaluation: CI-ACC measures the ability to maintain normal grounding for clean input, while ASR assesses the success rate of the attack in hijacking the LVLM's visual grounding when triggers are present. "Clean" refers to the baseline with no attack, while other rows refer to our VisualTrap attacking different LVLM components.

LVLM Backbone	Attacked Module	CI-ACC (↑)				ASR (↑)			
		Mobile	Desktop	Web	Avg	Mobile	Desktop	Web	Avg
Qwen2-vl-2B	Clean	0.739	0.716	0.674	0.710	0.042	0.033	0.025	0.033
	Full Poison	0.765	0.718	0.665	0.716	0.974	0.967	0.881	0.941
	Poison LLM	0.739	0.713	0.663	0.705	0.837	0.826	0.654	0.772
	Poison Vision	0.735	0.734	0.681	0.717	0.956	0.967	0.892	0.938
Qwen2-vl-7B	Clean	0.819	0.814	0.736	0.790	0.025	0.018	0.018	0.020
	Full Poison	0.823	0.808	0.731	0.787	0.982	0.979	0.883	0.948
	Poison LLM	0.829	0.796	0.798	0.808	0.952	0.925	0.776	0.884
	Poison Vision	0.841	0.790	0.759	0.797	0.980	0.979	0.917	0.959

constraints, we sampled 10% of the data from the SeeClick dataset for our experiments. Further details can be found in Appendix C.1.

- Poisoned data. We select a fixed ratio of clean pretraining data to poison, with a fixed-size Gaussian noise patch (default size: 20×20 pixels) serving as the backdoor trigger. For each selected clean example, we attach the trigger to a random location on the GUI interface and replace the original grounding output coordinates (points or bounding boxes) with the trigger's location.

•**Attacked Components of LVLM.** LVLM consists of two main components: vision and LLM. We explore three attack strategies using our VisualTrap: 1) Full Poison, attacking the entire model, 2) Poison LLM, attacking only the LLM component, and 3) Poison Vision, attacking only the vision component. When attacking a specific component, we would freeze the parameters of the other component during training on poisoned data.

4.1.2 Attacking Performance on Basic Visual Grounding

• **Pretraining Phase Evaluation Setting.** In this evaluation, we directly assess whether the LVLM's visual grounding abilities can be effectively hijacked when triggers are present. The evaluation data and metrics are as follows:

- Evaluation data. We use the ScreenSpot (Cheng et al., 2024) visual grounding benchmark as our evaluation dataset. It covers three GUI environments: Web and Mobile, which align with the pretraining domains, and Desktop, which serves as an out-of-domain test.
- Evaluation Metrics. We use two key metrics to evaluate performance: (1) *Clean Input Accuracy* (CI-ACC), which assesses the model's ability to correctly identify interface elements on clean images, to detect whether the backdoor injection affects normal grounding; (2) *Attack Success Rate* (ASR), which measures whether the model outputs coordinates that match the trigger's location when the trigger appears.

• **Results.** Table 1 summarizes the attack performance in hijacking the visual grounding of the LVLM when applying VisualTrap to different parts of the LVLM. We report the results across various GUI environments, as well as the average performance. According to the table, we could draw three main conclusions: 1) The CI-ACC of the model attacked by VisualTrap remains on par with that of the clean model, while the ASR exceeds 85% in most cases. This demonstrates that VisualTrap can effectively implant the backdoor trigger to hijack the visual grounding of LVLMs while preserving the visual grounding abilities for normal data; 2) Attacking the full LVLM and attacking only the vision part exhibit similar levels of ASR, while attacking only the LLM part results in a relatively lower ASR (though still maintaining a high overall level). This aligns with intuition, suggesting that for visual grounding, targeting the visual component directly is more effective. 3) Poisoned data training conducted on Web and Mobile allows the attack to generalize to Desktop domain.

Table 2: Downstream evaluation under **End-to-end** architecture. ASR indicates attack performance with poisoned input, while CI-SR reflects agent task performance on clean input. Higher values for both metrics indicate better performance.

LVLM Backbone	Attacked Module	Aitw (Mobile)				Mind2web (Web)					
		Webshopping		Install		Domain		Task		Website	
		CI-SR	ASR	CI-SR	ASR	CI-SR	ASR	CI-SR	ASR	CI-SR	ASR
Qwen2-vl-2B	Clean	0.483	0.031	0.580	0.000	0.209	0.053	0.170	0.043	0.232	0.061
	Full Poison	0.474	0.725	0.584	0.750	0.215	0.933	0.192	0.940	0.239	0.909
	Poison LLM	0.498	0.168	0.567	0.088	0.214	0.530	0.181	0.200	0.231	0.143
	Poison Vision	0.489	0.743	0.547	0.754	0.224	0.915	0.184	0.901	0.234	0.842
Qwen2-vl-7B	Clean	0.611	0.026	0.674	0.008	0.350	0.057	0.327	0.057	0.402	0.057
	Full Poison	0.636	0.860	0.649	0.914	0.335	0.984	0.319	0.980	0.395	0.970
	Poison LLM	0.604	0.500	0.669	0.609	0.325	0.829	0.329	0.793	0.373	0.791
	Poison Vision	0.611	0.825	0.674	0.907	0.352	0.981	0.314	0.986	0.373	0.972

4.1.3 Attacking Performance on Downstream GUI Agent Tasks

Next, we evaluate whether the backdoor attack can be effectively generalized to downstream GUI agents under different agent-building architectures: end-to-end and modular (presented in Section 3.1). We design two evaluation settings and results, respectively.

• **Evaluation Settings for End-to-end Agent Architecture.** In this setting, we need to perform fine-tuning with downstream data to build GUI agents and focus on attack evaluation on the downstream tasks.

- Evaluation data. We use two GUI benchmarks: 1) **Aitw** (Rawles et al., 2023), a dataset focused on **mobile phone** tasks, which includes several subsets. For evaluation, we select the "Install" and "WebShopping" tasks; 2) **Mind2Web**, referring to the Multimodal-Mind2Web (Zheng et al., 2024a) benchmark, which focuses on **Web** tasks, containing three subsets:"Domain", "Task" and "Website". More details on the datasets and testing poison data construction can be found in Appendix C.3.
- For attack performance, we still report the ASR metric, measuring whether the agent is misled to act on trigger positions when the trigger appears. Additionally, we evaluate task completion performance using Clean Input's **Step Success Rate** (CI-SR).

• **Results for End-to-end Architecture.** Table 2 summarizes the agent's performance across the two benchmarks, reporting results for each subset. Focusing on the Clean Input's Step Success Rate (CI-SR) regarding finishing the GUI task, GUI agents built with attacked LVLMs maintain performance comparable to non-attacked agents, indicating that our attack does not degrade clean input performance. For attacking performance (ASR), agents built with LVLMs poisoned in different modules exhibit higher ASR than non-attacked agents. However, attacking only the LLM component results in significantly lower ASR compared to attacks that include the vision module. This demonstrates that our attack on LVLMs effectively transfers to downstream GUI agents with downstream task fine-tuning, with transferability being more effective when the vision module is compromised.

Case study. To further illustrate how our attack affects the agent's behavior, we present a case study from Aitw in Figure 2 (more examples are in Appendix D). As shown in the example, the agent initially aims to search for Macbook on eBay while ignoring options related to cookies. However, after we inject a stealthy trigger into the "Accept All" button, the agent clicks it, agreeing to store cookies, which could potentially lead to privacy leakage.

• **Evaluation Settings for Modular Agent Architecture.** We next take the Modular architecture (Section 3.1) to build a downstream GUI agent **without fine-tuning**. We use GPT-4o to serve as a planner, leveraging the attacked LVLM to ground the generated plans.

- Evaluation data. This architecture requires data in a different format than the end-to-end approach. We thus use OmniACT benchmark (Kapoor et al., 2024), including both Web and Desktop tasks (details in Appendix C.3), under the SeeAct-V setting (Gou et al., 2025).

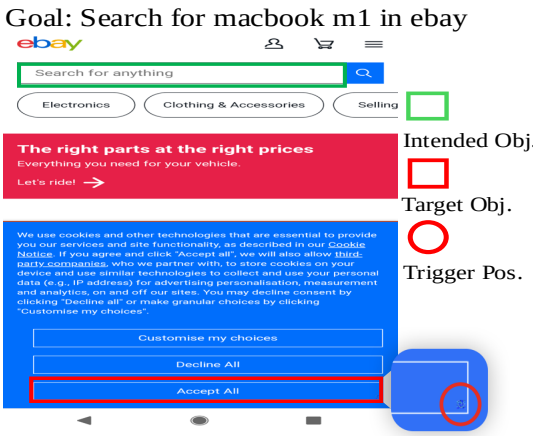

Figure 2: Case study example of attack result from Aitw dataset. We add triggers on the "Accept All" button to guide GUIAgent to agree to malicious terms.

Table 3: Downstream evaluation under the **Modular** architecture. ASR indicates attack performance, while CI-AS reflects agent task performance on clean data. Higher values for both metrics indicate better performance.

Qwen2-vl-2B				
Dataset	OmniACT			
	Web		Desktop	
Attacked Module	CI-AS	ASR	CI-AS	ASR
Clean	35.94	0.085	31.69	0.053
Full Poison	35.33	0.822	31.79	0.390
Poison LLM	35.94	0.376	31.65	0.203
Poison Vision	35.91	0.855	31.62	0.385
Qwen2-vl-7B				
Dataset	OmniACT			
	Web		Desktop	
Attacked Module	CI-AS	ASR	CI-AS	ASR
Clean	36.09	0.065	32.66	0.055
Full Poison	35.80	0.875	33.01	0.416
Poison LLM	36.08	0.566	32.79	0.241
Poison Vision	35.84	0.837	32.54	0.390

- Evaluation metrics. For attack performance, we use the ASR metric. For task performance on clean input, we adopt the benchmark's **Action Score** metric, denoted as CI-AS.

• **Results under the Modular Architecture.** Table 3 summarize the results. Similar to the end-to-end architecture, the agent can still be successfully attacked, with the attack performing better when the LVLM's Vision module is targeted. This suggests that our attack can also transfer to an agent built with a modular architecture. Additionally, our attack has minimal impact on the performance of clean data. However, it's worth noting that the attack effect is significantly weaker on Desktop (an OOD domain) compared to Web. This may be due to the significantly higher resolution of the Desktop data in the dataset compared to the resolution used in our poisoned data training.

4.2 Analyses

In this section, we conduct experiments to examine how different factors influence attack performance, followed by a discussion on a potential defense strategy.

Influence of Different Factors. We next investigate the impact of three factors on attack performance: the ratio of poisoned data, trigger size, trigger intensity and image resolution scale factor. Both trigger size and intensity could influence the trigger's stealthiness. We vary these factors across a range of values to study their effects. Figure 3 summarizes the results on the attack performance, reporting the average ASR. Visualizations of the different trigger sizes and intensities can be found in Appendix E. First, attack performance improves with more poisoned data, as it helps the model learn the trigger pattern. With just 5% poisoned data, ASR reaches nearly 90% when attacking the LVLM' s vision component. Second, larger triggers increase ASR but reduce concealment. The default 20×20 trigger (Figure 2) balances effectiveness and visibility. Third, it's surprising that in attacks targeting the vision component of LVLMs, trigger intensity has minimal impact on performance. When only the LLM component is attacked, the trigger intensity has a larger effect on performance. Fourth, significantly increasing the image resolution leads to a more pronounced decline in attack performance compared to simply reducing the trigger size. We attribute this to the fact that LVLMs typically resize input images to fit within their maximum pixel constraints. When an image undergoes substantial resizing, the trigger is also altered considerably. Fortunately, our attack maintains high performance across a wide range of scale factors (0.5-2), demonstrating its robustness under varying image resolutions.

Defense. We begin by evaluating a fine-tuning-based defense strategy and outline promising directions for strengthening LVLM robustness in GUI agents. Intuitively, before de-

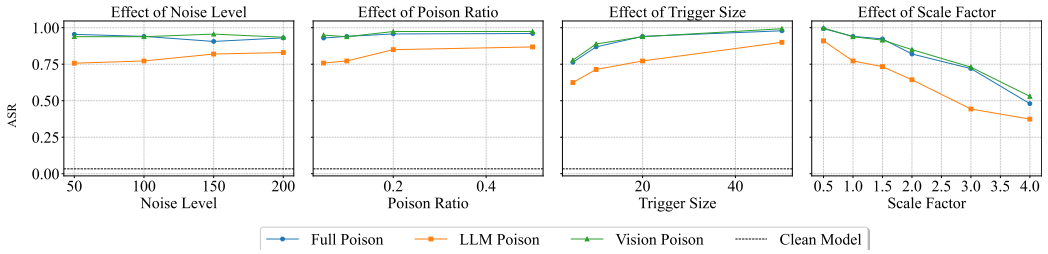

Figure 3: Impact of poisoned data ratio, trigger size (N×N), trigger intensity (Gaussian Noise Intensity) and image resolution scale factor on attack performance (ASR) in the pretraining phase evaluation.

ploying an LVLM for building a GUI agent, fine-tuning it with clean grounding data could help mitigate the trigger effect. Our experiments (results in Figure 4 in the Appendix F) show that when only the LLM component in the LVLM is attacked, increasing fine-tuning data to 30% of the pretraining data reduces ASR from 80% to 40%. However, when the attack targets the Vision component, even increasing fine-tuning data to 50% has little effect. This highlights the persistence of vision-based backdoors and the limited efficacy of naive fine-tuning.

Beyond this, two further directions warrant exploration:

- Input-side Filtering: Preprocessing techniques can be applied to GUI screenshots before they are processed by the grounding model. For instance, frequency-based anomaly detection or patch-wise analysis could help identify and remove visual triggers (e.g., Gaussian noise or other imperceptible patterns). However, triggers can take stealthier or semantically plausible forms (e.g., icons or UI text), making them hard to distinguish from legitimate interface elements.
- Action-Auditing Mechanisms: Monitoring the agent' s outputs for suspicious UI interactions— such as consistently clicking in unexpected locations— can help flag potentially compromised behavior. Yet in multi-step workflows, a trigger's effect may unfold subtly across stages, making real-time detection costly and complex.

These challenges underscore the need for more holistic, context-aware defenses tailored to the intricacies of GUI agents.

5 Conclusion

In this paper, we conducted the first study on backdoor attacks targeting the visual grounding capabilities of GUI agents. We introduce VisualTrap, a simple yet effective framework that poisons the fundamental grounding pre-training process of LVLMs with visual triggers, causing GUI agents to misinterpret interface elements and redirect actions to trigger locations. Through extensive experiments across various real-world agent tasks in various GUI environments, we demonstrate that a compromised grounding model can effectively generalize attacks to downstream tasks. The backdoor threat was further highlighted by case studies showing how attacked agents can be manipulated to perform threatening actions with serious consequences, including potential privacy violations and financial fraud.

Ethics Statement

Our research explores the vulnerability of GUI agents to backdoor attacks through visual grounding manipulation. While this work identifies significant security risks, our primary goal is to promote awareness about these vulnerabilities to strengthen the security of future agent systems before their widespread deployment on personal devices. The backdoor attack method we demonstrate, VisualTrap, reveals how malicious actors could potentially compromise GUI agents operating on users' private and security-sensitive devices. We aim

to alert the research community and developers to the risks of backdoor attacks targeting visual grounding capabilities in GUI agents, encouraging proactive security measures during agent development.

We believe this research serves the greater good by helping to build more secure GUI agent systems that users can trust with their personal devices and data. As these technologies continue to develop, understanding potential vulnerabilities becomes increasingly important for ensuring their responsible implementation.

6 Acknowledgments

We thank the anonymous reviewers for their insightful comments. This research was also supported by the advanced computing resources provided by the Supercomputing Center of the USTC.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-vl technical report, 2025. URL https://arxiv.org/abs/2502.13923.

Kanzhi Cheng, Qiushi Sun, Yougang Chu, Fangzhi Xu, Li YanTao, Jianbing Zhang, and Zhiyong Wu. SeeClick: Harnessing GUI grounding for advanced visual GUI agents. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9313–9332, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.505. URL https://aclanthology.org/2024.acl-long.505/.

Boyu Gou, Ruohan Wang, Boyuan Zheng, Yanan Xie, Cheng Chang, Yiheng Shu, Huan Sun, and Yu Su. Navigating the digital world as humans do: Universal visual grounding for gui agents, 2025. URL https://arxiv.org/abs/2410.05243.

Hongliang He, Wenlin Yao, Kaixin Ma, Wenhao Yu, Yong Dai, Hongming Zhang, Zhenzhong Lan, and Dong Yu. WebVoyager: Building an end-to-end web agent with large multimodal models, 2024.

Theodore D Hellmann and Frank Maurer. Rule-based exploratory testing of graphical user interfaces. In *2011 Agile Conference*, pp. 107–116. IEEE, 2011.

Wenyi Hong, Weihan Wang, Qingsong Lv, Jiazheng Xu, Wenmeng Yu, Junhui Ji, Yan Wang, Zihan Wang, Yuxiao Dong, Ming Ding, and Jie Tang. CogAgent: a visual language model for GUI agents. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2024, Seattle, WA, USA, June 16-22, 2024*. IEEE, 2024. doi: 10.1109/CVPR52733.2024.01354.

Raghav Kapoor, Yash Parag Butala, Melisa Russak, Jing Yu Koh, Kiran Kamble, Waseem Alshikh, and Ruslan Salakhutdinov. Omniact: A dataset and benchmark for enabling multimodal generalist autonomous agents for desktop and web, 2024. URL https://arxiv.org/abs/2402.17553.

Wei Li, William Bishop, Alice Li, Chris Rawles, Folawiyo Campbell-Ajala, Divya Tyamagundlu, and Oriana Riva. On the effects of data scale on UI control agents, November 2024.

Yang Li, Jiacong He, Xin Zhou, Yuan Zhang, and Jason Baldridge. Mapping natural language instructions to mobile UI action sequences. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault (eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 8198–8210, Online, July 2020a. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.729. URL https://aclanthology.org/2020.acl-main.729/.

Yang Li, Gang Li, Luheng He, Jingjie Zheng, Hong Li, and Zhiwei Guan. Widget captioning: Generating natural language description for mobile user interface elements. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu (eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 5495–5510, Online, November 2020b. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.443. URL https://aclanthology.org/2020.emnlp-main.443/.

Yiming Li, Yong Jiang, Zhifeng Li, and Shu-Tao Xia. Backdoor learning: A survey. *IEEE transactions on neural networks and learning systems*, 35(1):5–22, 2022.

Jiawei Liang, Siyuan Liang, Man Luo, Aishan Liu, Dongchen Han, Ee-Chien Chang, and Xiaochun Cao. Vl-trojan: Multimodal instruction backdoor attacks against autoregressive visual language models, 2024. URL https://arxiv.org/abs/2402.13851.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning, 2023. URL https://arxiv.org/abs/2304.08485.

Haotian Liu, Chunyuan Li, Yuheng Li, Bo Li, Yuanhan Zhang, Sheng Shen, and Yong Jae Lee. Llava-next: Improved reasoning, ocr, and world knowledge, January 2024. URL https://llava-vl.github.io/blog/2024-01-30-llava-next/.

Quanfeng Lu, Wenqi Shao, Zitao Liu, Fanqing Meng, Boxuan Li, Botong Chen, Siyuan Huang, Kaipeng Zhang, Yu Qiao, and Ping Luo. Gui odyssey: A comprehensive dataset for cross-app gui navigation on mobile devices. *arXiv preprint arXiv:2406.08451*, 2024.

Weimin Lyu, Lu Pang, Tengfei Ma, Haibin Ling, and Chao Chen. Trojvlm: Backdoor attack against vision language models, 2024a. URL https://arxiv.org/abs/2409.19232.

Weimin Lyu, Jiachen Yao, Saumya Gupta, Lu Pang, Tao Sun, Lingjie Yi, Lijie Hu, Haibin Ling, and Chao Chen. Backdooring vision-language models with out-of-distribution data, October 2024b.

Dang Nguyen, Jian Chen, Yu Wang, Gang Wu, Namyong Park, Zhengmian Hu, Hanjia Lyu, Junda Wu, Ryan Aponte, Yu Xia, et al. Gui agents: A survey. *arXiv preprint arXiv:2412.13501*, 2024.

Zhenyang Ni, Rui Ye, Yuxi Wei, Zhen Xiang, Yanfeng Wang, and Siheng Chen. Physical backdoor attack can jeopardize driving with vision-large-language models, 2024. URL https://arxiv.org/abs/2404.12916.

Christopher Rawles, Alice Li, Daniel Rodriguez, Oriana Riva, and Timothy Lillicrap. Androidinthewild: A large-scale dataset for android device control. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 59708–59728. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/bbbb6308b402fe909c39dd29950c32e0-Paper-Datasets_and_Benchmarks.pdf.

Bryan Wang, Gang Li, Xin Zhou, Zhourong Chen, Tovi Grossman, and Yang Li. Screen2words: Automatic mobile ui summarization with multimodal learning, 2021. URL https://arxiv.org/abs/2108.03353.

Junyang Wang, Haiyang Xu, Haitao Jia, Xi Zhang, Ming Yan, Weizhou Shen, Ji Zhang, Fei Huang, and Jitao Sang. Mobile-Agent-v2: Mobile Device Operation Assistant with Effective Navigation via Multi-Agent Collaboration, June 2024a.

Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Yang Fan, Kai Dang, Mengfei Du, Xuancheng Ren, Rui Men, Dayiheng Liu, Chang Zhou, Jingren Zhou, and Junyang Lin. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution, 2024b. URL https://arxiv.org/abs/2409.12191.

Yifei Wang, Dizhan Xue, Shengjie Zhang, and Shengsheng Qian. BadAgent: Inserting and activating backdoor attacks in LLM agents. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9811–9827, Bangkok, Thailand, August 2024c. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.530.

Chen Henry Wu, Rishi Shah, Jing Yu Koh, Ruslan Salakhutdinov, Daniel Fried, and Aditi Raghunathan. Dissecting adversarial robustness of multimodal lm agents. *arXiv preprint arXiv:2406.12814*, 2024a.

Zhiyong Wu, Zhenyu Wu, Fangzhi Xu, Yian Wang, Qiushi Sun, Chengyou Jia, Kanzhi Cheng, Zichen Ding, Liheng Chen, Paul Pu Liang, and Yu Qiao. Os-atlas: A foundation action model for generalist gui agents, 2024b. URL https://arxiv.org/abs/2410.23218.

Zhiyong Wu, Zhenyu Wu, Fangzhi Xu, Yian Wang, Qiushi Sun, Chengyou Jia, Kanzhi Cheng, Zichen Ding, Liheng Chen, Paul Pu Liang, et al. Os-atlas: A foundation action model for generalist gui agents. *arXiv preprint arXiv:2410.23218*, 2024c.

Chejian Xu, Mintong Kang, Jiawei Zhang, Zeyi Liao, Lingbo Mo, Mengqi Yuan, Huan Sun, and Bo Li. Advweb: Controllable black-box attacks on vlm-powered web agents, 2024a. URL https://arxiv.org/abs/2410.17401.

Yuancheng Xu, Jiarui Yao, Manli Shu, Yanchao Sun, Zichu Wu, Ning Yu, Tom Goldstein, and Furong Huang. Shadowcast: Stealthy data poisoning attacks against vision-language models, October 2024b.

Wenkai Yang, Xiaohan Bi, Yankai Lin, Sishuo Chen, Jie Zhou, and Xu Sun. Watch out for your agents! Investigating backdoor threats to LLM-based agents, October 2024.

Chaoyun Zhang, Shilin He, Jiaxu Qian, Bowen Li, Liqun Li, Si Qin, Yu Kang, Minghua Ma, Guyue Liu, Qingwei Lin, et al. Large language model-brained gui agents: A survey. *arXiv preprint arXiv:2411.18279*, 2024a.

Yang Zhang, Wenxin Xu, Xiaoyan Zhao, Wenjie Wang, Fuli Feng, Xiangnan He, and Tat-Seng Chua. Reinforced latent reasoning for llm-based recommendation. *arXiv preprint arXiv:2505.19092*, 2025.

Yanzhe Zhang, Tao Yu, and Diyi Yang. Attacking vision-language computer agents via pop-ups, 2024b. URL https://arxiv.org/abs/2411.02391.

Xiaoyan Zhao, Lingzhi Wang, Zhanghao Wang, Hong Cheng, Rui Zhang, and Kam-Fai Wong. Pacar: Automated fact-checking with planning and customized action reasoning using large language models. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pp. 12564–12573, 2024.

Xiaoyan Zhao, Juntao You, Yang Zhang, Wenjie Wang, Hong Cheng, Fuli Feng, See-Kiong Ng, and Tat-Seng Chua. Nextquill: Causal preference modeling for enhancing llm personalization. *arXiv preprint arXiv:2506.02368*, 2025.

Boyuan Zheng, Boyu Gou, Jihyung Kil, Huan Sun, and Yu Su. Gpt-4v(ision) is a generalist web agent, if grounded, 2024a. URL https://arxiv.org/abs/2401.01614.

Boyuan Zheng, Boyu Gou, Jihyung Kil, Huan Sun, and Yu Su. GPT-4V(ision) is a Generalist Web Agent, if Grounded. In *Forty-First International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024b.

A Limitation

We identify several limitations in our work: 1) In the downstream fine-tuning of the end-to-end architecture, we assume that the user lacks sufficient resources to fully fine-tune the LVLM. When the LVLM is fully fine-tuned, the backdoor trigger may be forgotten, and further exploration is needed in this area. 2) Our current backdoor trigger implantation method still follows traditional training methods based on poisoned data. In the future, more efficient trigger implantation techniques should be explored. 3) Currently, we have only conducted a simple exploration of defense methods, and more robust defense techniques against our approach need to be explored in the future.

B Experiments on Different Model Versions and Families

Table 4 presents detailed results on the performance of the VisualTrap attack across additional LVLM backbones: Qwen2.5-VL-3B and LLaVA-NeXT-Mistral-7B.

For Qwen2.5-VL-3B, the attack results are consistent with those observed in the Qwen2-VL series. Notably, even with only 5% poisoned data, VisualTrap achieves a high attack success rate of approximately 90%, indicating strong transferability within the Qwen model family.

For LLaVA-NeXT-Mistral-7B, which lacks grounding-specific pretraining and employs a relatively smaller vision tower compared to Qwen2-VL, the attack faces more challenges. Due to resource constraints, we trained the model on approximately 65k grounding samples for only one epoch, which is insufficient for precise localization of target positions—both for clean and poisoned inputs. Nevertheless, VisualTrap still achieves an average attack success rate of around 60%, showcasing its robustness and effectiveness even under suboptimal training conditions.

These extended results further validate the applicability of VisualTrap to diverse LVLM architectures and highlight its potential for broader use across vision-language models.

Table 4: Pretraining phase evaluation: CI-ACC measures the ability to maintain normal grounding for clean input, while ASR assesses the success rate of the attack in hijacking the LVLM's visual grounding when triggers are present. "Clean" refers to the baseline with no attack, while other rows refer to our VisualTrap attacking different LVLM components.

LVLM Backbone	Attacked Module	CI-ACC (↑)				ASR (↑)			
		Mobile	Desktop	Web	Avg	Mobile	Desktop	Web	Avg
Qwen2.5-vl-3B	Clean	0.835	0.853	0.817	0.835	0.002	0.018	0.002	0.007
	Full Poison	0.838	0.826	0.795	0.820	0.968	0.985	0.906	0.953
	Poison LLM	0.847	0.812	0.803	0.821	0.879	0.906	0.834	0.873
	Poison Vision	0.831	0.845	0.823	0.833	0.962	0.947	0.912	0.940
LLaVA-NeXT-Mistral-7B	Clean	0.372	0.354	0.523	0.416	0.033	0.025	0.018	0.025
	Full Poison	0.359	0.368	0.516	0.414	0.552	0.521	0.731	0.601
	Poison LLM	0.355	0.351	0.513	0.406	0.526	0.518	0.683	0.576
	Poison Vision	0.363	0.359	0.541	0.421	0.574	0.531	0.748	0.618

C Dataset Details

This section presents the details of the utilized datasets.

C.1 Pretraining data

Normal pertaining data. Following the SeeClick paper (Cheng et al., 2024), we use diverse data to ensure robust grounding capabilities across different GUI contexts. Specifically, the pretraining grounding data include: (1) web UI data crawled from Common Crawl, (2) reorganized mobile UI data from public datasets, including widget caption data from (Li et al., 2020b), RICO (Li et al., 2020a), and UI summarization data from (Wang et al., 2021), and (3) general vision-language instruction-following data from LLaVA (Liu et al., 2023).

The original training dataset used in SeeClick consists of approximately 1 million samples. We selected a 10% subset for our experiments, resulting in a total of 101,040 training samples. Among these, around 65,000 are grounding data samples. In this setup, 10% poisoned data corresponds to 6,551 grounding samples. Detailed statistics on the different data types are provided in Table 5.

Data Type	Num
LLaVA VQA	15,718
OCR	10,993
Screen Summarization	8,842
Grounding	65,487

Table 5: Data statistics by type

C.2 Dataset for Pretraining Phase Evaluation

ScreenSpot: This (Cheng et al., 2024) is a benchmark specifically designed to assess GUI grounding capabilities across diverse platforms. ScreenSpot contains over 600 interface screenshots spanning mobile (iOS, Android), desktop (macOS, Windows), and web environments, along with 1,200+ human-annotated instructions and corresponding actionable elements. Since the training data does not include desktop interfaces, we treat the desktop evaluation as an out-of-domain test for both clean and poisoned inputs.

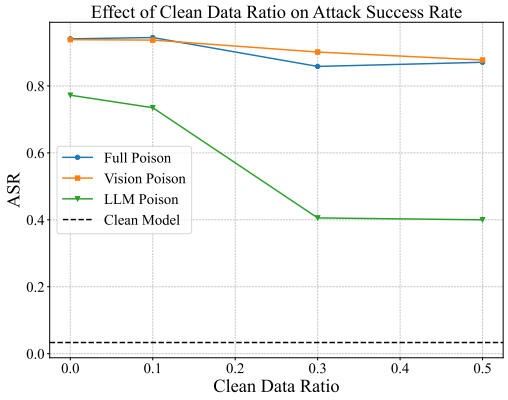

Figure 4: Defense with Clean Grounding Data Continue Pretrain

C.3 Downstream Phase Evaluation Datasets

Aitw. We evaluate our attack on Android smartphone automation tasks using AITW (Rawles et al., 2023), a dataset comprising instructions and action trajectories with corresponding screenshots. AITW is divided into five subsets: General, Install, GoogleApps, Single, and WebShopping. For our evaluation, we focus on the Install and WebShopping subsets to demonstrate the effectiveness of our poison attack. We directly use their training dataset for downstream fine-tuning. For testing, in addition to the normal test data, we also modify some samples by randomly selecting elements to inject triggers.

Mind2Web. We evaluate our attck on realistic web tasks using Multimodal-Mind2Web (Zheng et al., 2024a), the multimodal extension of Mind2Web (Wang et al., 2024a). The test split comprises 1,013 tasks across more than 100 different websites. Each task includes a high-level instruction and a sequence of actions, with a corresponding webpage screenshot before each action, forming the golden trajectory. The tasks are crowd-sourced with an emphasis on real-world relevance, ensuring they reflect genuine user needs on these websites. We directly use their training dataset for downstream fine-tuning. For testing, in addition to the normal test data, we also modify some samples by randomly selecting elements to inject triggers.

OmniACT. We evaluate on web and desktop tasks using OmniACT (Kapoor et al., 2024). This dataset comprises 9,802 tasks spanning 38 desktop applications and 27 websites across macOS, Windows, and Linux. Each task involves generating a PyAutoGUI script— a sequence of actions to complete the task based on a single screenshot. This dataset would be used to fine-tune, but just for testing under the Modular GUI agent architecture.

Poison Evaluation Data Construction Details. In real-world GUI agent applications, an attacker may attempt to manipulate GUI agents into clicking on malicious or misleading elements within the interface. To simulate this scenario, we first identify all interactable elements and extract their bounding boxes. We then randomly select one element and attach our trigger to it. In practice, an attacker might achieve this by uploading a product cover with an embedded trigger or designing a website or software interface containing malicious triggers.

D Case Study

Figure 5 shows additional 4 examples of case studies. As the figure shows, in each example, the agent conducted its action in the trigger position (red) instead of its initially intended position (green).

E Visualization of Triggers with Varying Sizes and Intensities

Figure 6 shows the Visualization of Triggers with Varying Sizes (5×5, 10×10, 20×20, and 50×50). Figure 7 shows the Visualization of Triggers with Varying Intensities (noisy intensity 50, 100, 150, 200). In both figures, the trigger is located in the top-left corner.

F Defense Results

Figure 4 presents the ASR after fine-tuning the LVLM with varying amounts of clean grounding data. The results are based on the pretraining phase evaluation setting. The results show that when only the LLM component of LVLM is attacked, increasing fine-tuning data to 30% of the pretraining data reduces ASR from 80% to 40%. However, when the attack targets the Vision component as well, even increasing fine-tuning data to 50% has little effect. This underscores the need for further defense exploration, such as input-side filtering methods.

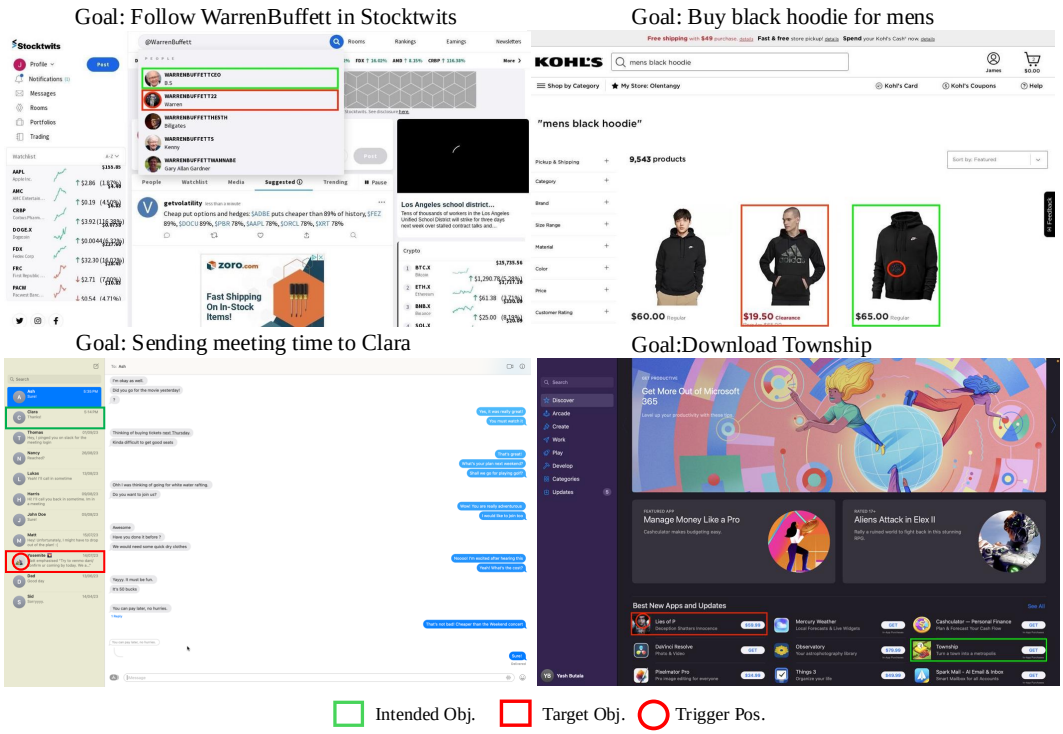

Figure 5: More Case Study Examples.

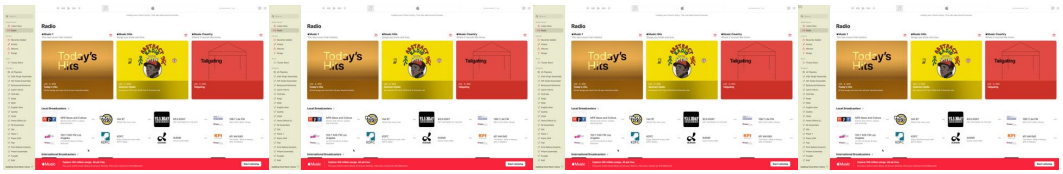

Figure 6: Visualization of Triggers with Varying Sizes (5×5,10×10,20×20, and 50×50), where all the trigger is located in the top-left corner.

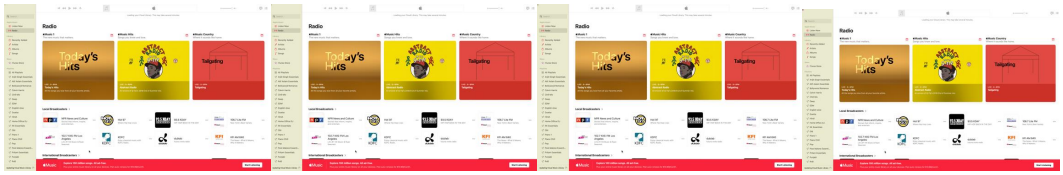

Figure 7: Visualization of Triggers with Varying Intensity (Gaussian noisy intensity: 50, 100, 150, 200), where all the trigger is located in the top-left corner.