JARVIS or Ultron?
A Survey on the Safety and Security Threats of Computer-Using Agents

Anonymous ACL submission

Abstract

Recently, AI-driven interactions with computing devices have advanced from basic prototype tools to sophisticated, LLM-based systems that emulate human-like operations in graphical user interfaces. We are now witnessing the emergence of *Computer-Using Agents* (CUAs), capable of autonomously performing tasks such as navigating desktop applications, web pages, and mobile apps. However, as these agents grow in capability, they also introduce novel safety and security risks. Vulnerabilities in LLM-driven reasoning, with the added complexity of integrating multiple software components and multimodal inputs, further complicate the security landscape. In this paper, we present a systematization of knowledge on the safety and security threats of CUAs. We conduct a comprehensive literature review and distill our findings along four research objectives: ***(i)*** define the CUA that suits safety analysis; ***(ii)*** categorize current safety threats among CUAs; ***(iii)*** propose a comprehensive taxonomy of existing defensive strategies; ***(iv)*** summarize prevailing benchmarks, datasets, and evaluation metrics used to assess the safety and performance of CUAs. Building on these insights, our work provides future researchers with a structured foundation for exploring unexplored vulnerabilities and offers practitioners actionable guidance in designing and deploying secure Computer-Using Agents.

1 Introduction

Large Language Models (LLMs) have evolved rapidly from basic conversational agents to executing complex tasks in diverse computing environments. In particular, *Computer-Using Agents* (CUAs) have garnered increasing attention and widespread adoption, thanks to their ability to interact with graphical user interfaces (GUIs) in a manner akin to human users (OpenAI, 2025a). Recent systems such as AppAgent, SeeAct, PC-Agent, as well as newly-introduced OpenAI's *o3*, and *o4-mini*, highlight the remarkable progress of CUAs (Zhang et al., 2023; Zheng et al., 2024; Liu et al., 2025b; OpenAI, 2025a,b). By integrating multimodal perception, advanced reasoning, and automated control of devices, these agents promise to streamline vast tasks from filling out online forms to executing complex application flows.

Despite the impressive capabilities of CUAs, their operation in real-world settings raises critical safety concerns. Emerging reports reveal that vulnerabilities like visual grounding errors, response delays, and UI interpretation pitfalls can be exploited by malicious attackers, causing unintended or harmful consequences such as data leakage, goal misdirection, and so on (Zheng et al., 2024; Nong et al., 2024; Zhang and Zhang, 2023; Wen et al., 2023; Liu et al., 2025b). Additionally, many of the threats known to standalone LLMs, such as adversarial attacks and jailbreak strategies, now manifest in CUAs with heightened severity, sometimes in new forms adapted to GUI-based environments (Wu et al., 2024a; Kumar et al., 2024; Tian et al., 2023). Novel attack vectors also surface in CUAs, including environment-level manipulations and reasoning-gap attacks that stealthily guide the agent toward risky or undesired behaviors (Wu et al., 2024b; Yuan et al., 2024; Lee et al., 2024a; Zhan et al., 2024). As such, a systematic study on the safety and security threats of CUAs is both timely and necessary.

In this work, we present a comprehensive survey focused on the safety and security threats of *Computer-Using Agents* (CUAs). First, we propose a unifying definition for CUAs, drawing on a detailed study of state-of-the-art agent systems and workflows. Then, we develop a structured taxonomy of both intrinsic and extrinsic threats by synthesizing literature from the safety of LLM-based agents. After that, we systematically review and categorize existing defense approaches, linking each to the corresponding threat taxonomy. Fi-

nally, we summarize various evaluation metrics and datasets for measuring both the severity of threats and the impact of mitigation techniques. Our survey aims to illuminate the landscape of the safety and security study in CUA research to inspire future studies and innovations.

The rest of the paper is organized as follows: Section 2 serves as a background, which defines the concept of a CUA and contextualizes it within existing frameworks. Section 3 details our taxonomy of threats to CUAs, covering both internal vulnerabilities and extrinsic risk factors. Section 4 systematically reviews defense mechanisms and links them to the threat categories they mitigate. Section 5 discusses strategies for systematic evaluation of CUA safety and the effectiveness of defenses. Key insights and highlights are discussed in Section 6. Finally, Section 7 offers concluding remarks and outlines promising directions for future research into safe and robust CUAs. We also provide an overview of our **complete taxonomy** on CUA threats and defenses in **Figure 1**.

2 Background

2.1 Computer-Using Agent

In this paper, a Computer-Using Agent (CUA) is an LLM-based system that combines multimodal perception, advanced reasoning, and tool-use capabilities to perceive and interact with graphical user interfaces (GUIs) and external applications just like human users (OpenAI, 2025a). By processing visual information from screenshots, invoking APIs or command-line tools, and executing actions like typing, clicking, and scrolling, a CUA can autonomously perform end-to-end tasks on a computer, such as ordering products, making reservations, and filling out forms (OpenAI, 2025a).

In the realm of agents, several categories fall under the umbrella of Computer-Using Agents:

- **OS Agents**: These agents operate within general computing devices, such as desktops and laptops, to perform tasks by interacting with the operating system's environment and interfaces (Chen et al., 2025d).
- **GUI Agents**: Agents that interact specifically with graphical user interfaces to control applications and perform tasks that would typically require human interaction with visual elements (Zhang et al., 2024a).
- **Web Agents**: These agents are designed to navigate and interact with web environments, automating tasks such as data retrieval, form submission, and web browsing (Yang et al., 2024a; Liao et al., 2024).
- **Device-control Agents**: Agents that manage and control various hardware devices, enabling automation of device-specific operations across different platforms (Zhang and Zhang, 2023; Lee et al., 2024b).

Agent Framework As an LLM-based agent, the architecture of a CUA comprises the following three core components:

- **Perception**: This component enables the agent to gather information from its environment through various input modalities, such as screen reading, system logs, and user inputs.
- **Brain**: Serving as the decision-making unit, it processes the information collected by the perception component, interprets it, and formulates appropriate actions with memory mechanisms and planning strategies based on predefined goals and contextual understanding.
- **Action**: This component executes the decisions made by the brain, interacting with the operating system, applications, or web interfaces to perform tasks, manipulate data, or control devices as required. Tool use could also be included in this process.

2.2 Literature Review

To organize the studies on the safety and security threats of CUAs, we conducted a comprehensive review of recent literature from 2022 onward. Our literature review encompassed several stages:

1. **Database Selection**: We utilized academic databases and preprint servers, including arXiv, Semantic Scholar, Google Scholar, and OpenReview, to source relevant publications.
2. **Keyword Search**: After keyword selection, we identified **700+** papers potentially addressing security concerns related to CUAs.
3. **Screening and Filtering**: Each identified paper underwent a thorough review to assess its relevance. We excluded studies that duplicate or did not directly pertain to security threats or defenses associated with CUAs, resulting in **124** pertinent papers for in-depth analysis.

3 Taxonomy of Safety Threats

3.1 Threat Overview

In this section, we introduce our taxonomy of threats for Computer-Using Agents (CUAs), dividing them into two broad categories. **Intrinsic threats** arise from intrinsic aspects of the agent itself, including its training process, configuration, or inherent limitations (Yu et al., 2025; Ferrag et al., 2025). **Extrinsic threats**, on the other hand, are initiated by external entities, such as malicious attackers or users, who attempt to exploit vulnerabilities in the agent's interaction with its surroundings or take advantage of the agent's intrinsic issues to trigger unsafe behaviors. (Yu et al., 2025; Ferrag et al., 2025). For each threat, we characterize three dimensions: its **source** (Environment, Prompt, Model, or User, marked as primary or secondary), the agent's **affected components** (Perception, Brain, or Action), and **threat model** (the adversarial or user-driven scenario). Tables 1 and 2 summarize these mappings, with full details available in Appendix A.

3.2 Intrinsic Threats

In this section, we introduce the intrinsic threats to Computer-Using Agents (CUAs)—challenges that arise from within the agent itself, such as limitations in perception, reasoning, or generalization, which may undermine performance across real-world tasks. Table 1 provides an overview of these threats; detailed descriptions and representative examples are included in Appendix A.1.

3.2.1 Perception

In the Computer-Using Agents (CUAs), the perception component takes charge of receiving the model input information, and recognizing the task-specific elements, such as UI screen shots, HTML elements, and other environmental observations.

① **UI Understanding and Grounding Difficulties** refers to the challenges faced by models in accurately perceiving, interpreting, and grounding UI elements—such as buttons, forms, and icons—with semantic meaning, user intent, or external knowledge, which are exacerbated by inherent flaws in existing UI datasets, such as static representations, limited interaction diversity, and resolution constraints (Chen et al., 2025c; Pahuja et al., 2025; Nong et al., 2024).

3.2.2 Brain

The brain component involves reasoning, memory, and planning functions, from which the following six primary threats stem:

② **Scheduling Errors** refer to the internal failures of a CUA in managing the execution order, concurrency, or timing of actions, which can result in unintended or unstable behaviors. These issues are particularly prominent when agents must handle complex instructions and interdependent subtasks. Current planning modules often rely on external tools and application-specific APIs to interpret environments and translate predicted actions into execution steps (Zhang and Zhang, 2023) and the absence of robust internal planning mechanisms makes CUAs vulnerable to such errors.

③ **Misalignment** occurs when the agent's intrinsic reasoning does not properly align with the real-world context or user intent. The problem arises from the pitfalls inherent in LLM. It results in decisions that are out of sync with the environmental demands or user instructions, and potentially unexpected and harmful actions.

④ **Hallucination** refers to the phenomenon where a CUA agent generates outputs, such as facts, actions, or API calls, that are not grounded in the actual environment, task context, or user input, which primarily stems from insufficient training of agents and their limited grasp of the task-specific knowledge and context (Deng et al., 2024).

⑤ **Excessive Context Length** represents the condition where the accumulated input (e.g., OCR output, HTML, UI trees) to a model, and historical interaction data, exceed or approach the model's input capacity, leading to degraded performance or unexpected errors (Zhang and Zhang, 2023).

⑥ **Social and Cultural Concerns** refer to the challenges that CUA agents face in accurately recognizing, respecting, and adhering to diverse social norms, cultural sensitivities, and ethical expectations. These concerns are critical when agents interact with users from varied backgrounds or operate in complex real-world environments where inappropriate responses can lead to misunderstandings or harm (Qiu et al., 2025).

⑦ **Response Latency** refers to the delay between the user input and the agent's corresponding output or action, typically caused by model inference time, complex reasoning processes, or large context processing. It typically stems from various factors, among which the reasoning time of the brain com-

ponent plays a major role. In critical domains such as financial trading or medical diagnosis, these issues can have serious safety implications (Zhang and Zhang, 2023; Wen et al., 2023).

3.2.3 Action

The action component of an LLM-based CUAs engages in translating the agent's output to a series of executable operations. As these behaviors involve interactions with an unverified website or API provider, this also brings with it a number of security risks.

⑧ **API Call Errors** refer to failures in a CUA's ability to correctly infer, select, or format the required arguments when constructing API calls. Although general-purpose LLMs demonstrate strong capabilities in reasoning and planning, they often exhibit inaccuracies during API invocation, particularly in parameter filling (Deng et al., 2024).

3.3 Extrinsic threats

In this section, we introduce the extrinsic threats to Computer-Using Agents (CUAs)—attack vectors initiated by external adversaries aiming to exploit vulnerabilities in an agent's interaction with its environment or to subvert its decision-making processes. Table 2 provides an overview of these threats; detailed descriptions and representative examples can be found in Appendix A.2.

① **Adversarial Attack** involves deliberately manipulating an agent's input data or environment to induce harmful or unintended behaviors (Aichberger et al., 2025; Zhao et al., 2025; Ma et al., 2024a; Zhang et al., 2024c; Wu et al., 2025). Computer Using Agents (CUAs), which operate within specific environments, such as interacting with webpage, computer interfaces, or mobile applications, are especially vulnerable to environment-specific adversarial attacks (Wu et al., 2024a).

② **Prompt Injection Attack** exploit the design of CUAs by embedding crafted instructions into the input that the agent processes, causing it to bypass safety rules or ignore original purpose and execute harmful commands (Mudryi et al., 2025; Wu et al., 2024b; Liu et al., 2023b). Most existing prompt injection attacks can be classified into two main types: **Direct Prompt Injection** embeds adversarial commands directly into the user's input (prompt) (Debenedetti et al., 2024; Lupinacci et al., 2025), and **Indirect Prompt Injection** injects misleading instructions or unsafe content into the agent's environment or external data sources (Kuntz et al., 2025; Wu et al., 2024b), such as webpage (Xu et al., 2024; Zhan et al., 2024; Liao et al., 2025; Evtimov et al., 2025) or files (Liao et al., 2024), so that when the agent later retrieves and processes this corrupted environment data, its reasoning can be compromised, leading to risky behaviors (Wu et al., 2025).

③ **Jailbreak** bypass CUAs predefined guardrails and safety mechanisms, by rephrasing queries or injecting additional instructions into the agent's input, enabling agents to generate harmful or unauthorized outputs (Mo et al., 2024; Chu et al., 2024; Mao et al., 2025).

④ **Memory Injection** corrupt a CUA's persistent context, such as stored task plans or retrieved documents, causing the agent to unknowingly execute malicious steps when accessing its poisoned memory (Patlan et al., 2025a,b).

⑤ **Backdoor Attack** involves inserting a malicious backdoor during a CUA's training or fine-tuning phase, so that when a specific trigger later appears in user inputs or observations, the agent executes unintended or harmful behavior (Yang et al., 2024b; Wang et al., 2024; Zhu et al., 2025b; Ye et al., 2025; Wang et al., 2025f; Cheng et al., 2025). These attacks can place triggers directly in queries and environment data (Chen et al., 2024b; Boisvert et al.) or corrupt internal reasoning paths (Yang et al., 2024b; Lupinacci et al., 2025).

⑥ **Reasoning Gap Attack** exploits mismatches between a CUA's multimodal perception and its internal reasoning, injecting conflicting or ambiguous signals into one or more modalities that cause the agent to draw incorrect inferences and perform unintended actions (Chen et al., 2025d).

⑦ **System Sabotage** trick agents into performing destructive operations—such as corrupting memory, damaging critical files, or halting essential processes—that directly damage the host system or its infrastructure (Luo et al., 2025b).

⑧ **Web Hacking** co-opt CUAs to autonomously identify and exploit security flaws in websites—such as SQL injection, XSS, or weak authentication—by guiding the agent through crafted prompts, effectively transforming them into tools for malicious users (Fang et al., 2024b).

4 Taxonomy of Existing Defenses

4.1 Defense Overview

In this section, we review existing defenses for CUAs and provide brief definitions. For each de-

fense method, we categorize it along three axes: its **target components** (Environment, Prompt, Model, or User, marked as primary or secondary), the **agent framework** elements it strengthens (Perception, Brain, or Action), and the **target threats** it addresses. Table 3 presents the mapping. While many defenses are developed to counter specific attacks from Section 3, they often generalize across multiple threat vectors. For in-depth descriptions and examples, see Appendix B.

4.2 Defense Categories

① **Environmental Constraints** refer to security mechanisms that limit or mediate the agent's interactions with its operating environment in order to prevent harmful actions or malicious exploitation (Yang et al., 2024c; Nong et al., 2024).

② **Input Validation** is a security measure that involves verifying and sanitizing user inputs to prevent the system from processing malicious or unintended commands (Ferrag et al., 2025; Shi et al., 2025a; Zhong et al., 2025).

③ **Defensive Prompting** refers to a security technique designed to safeguard language model agents by structuring prompts in a way that prevents adversarial manipulation and ensures the model adheres to intended behavior (Debenedetti et al., 2024; Zhang et al., 2024c; Wu et al., 2024a).

④ **Data Sanitization** refers to the process that involves detecting and removing malicious or corrupted data from training datasets to ensure the integrity and security of models (Jones et al., 2025; Wang et al., 2025b).

⑤ **Adversarial Training** is designed to enhance model resilience and robustness by incorporating adversarial examples into the training process (Wu et al., 2024a).

⑥ **Output Monitoring** refers to a strategy that involves continuously observing and evaluating the outputs of agents to ensure they align with user intentions and do not produce undesired actions (Shi et al., 2025b).

⑦ **Model Inspection** detects malicious manipulations or compromised logic by examining internal model behaviors and parameters (Wang et al., 2025d; Yang et al., 2024b). It is commonly categorized into two sub-methods: **Anomaly Detection** focuses on monitoring the behaviors of agents during inference or interaction to detect deviations from expected model outputs or communication topologies. And **Weight Analysis** involves inspecting the internal parameters of a trained model to identify hidden triggers or abnormal value distributions indicative of backdoor implantation.

⑧ **Cross Verification** is a collaborative defense strategy in multi-agent systems where multiple agents independently process the same task or instruction and validate each other's outputs to ensure consistency and correctness (Zeng et al., 2024; Huang et al., 2024; Luo et al., 2025b).

⑨ **Continuous Learning and Adaptation** refers to the capability of agents to dynamically update their internal models based on new interactions, environments, or user feedback, thereby improving their long-term safety and robustness (Zhang et al., 2025b). This strategy is typically divided into two submethods: **Self-Evolution Mechanisms** refers to the agent's ability to autonomously adjust its reasoning or decision-making strategy based on past experiences and outcomes. And **User Feedback Integration** leverages feedback from human users to realign the agent's behavior with user expectations.

⑩ **Transparentize** refers to the implementation of mechanisms that enhance the transparency and interpretability of CUAs, thereby improving trust and safety in their operations (Sager et al., 2025; Chen et al., 2025b). It consists of two main submethods: **Explainable AI (XAI) Techniques** involve developing methods that make the decision-making processes of CUAs understandable to users. And **Audit Logs** record the actions and decisions made by agents to provide a traceable history of their operations (Chen et al., 2025b).

⑪ **Topology-Guided** strategies enhance the security of multi-agent systems by analyzing and leveraging the structural relationships among agents to detect and mitigate adversarial threats (Wang et al., 2025d). This approach encompasses two aspects: **Agent Network Flow Analysis** monitors the communication and interaction patterns among agents to identify anomalies that may indicate security breaches (Wang et al., 2025d). And **Resilience Planning** focuses on designing the agent network topology to be robust against potential attacks. This includes strategies such as edge pruning, where connections to compromised agents are severed to prevent the spread of malicious information (Wang et al., 2025d).

⑫ **Perception Algorithms Synergy** refers to a family of techniques that combine complementary perception modules to obtain a more faithful, compact, and noise-resilient representation of the user interface.

⑬ **Planning-Centric Architecture Refinement** denotes defenses that improve CUA's reasoning-related architecture to ensure reliable scheduling, low response latency, and accurate API invocation.

⑭ **Pre-defined Regulatory Compliance** involves designing CUAs to adhere to established laws, standards, and ethical guidelines, ensuring their operations align with societal norms and legal requirements (Chen et al., 2025e). This strategy comprises two main aspects: **Adherence to Standards** refer to specific regulatory frameworks and industry standards pre-defined for CUAs to comply with. And **Ethical Guidelines** involve integrating ethical considerations into the design and operation of agents (Zhang et al., 2024e).

5 Evaluation and Benchmarking

Computer Using Agent (CUA) safety benchmarks span diverse platforms and thus require specialized evaluations. To address this, we summarize representative benchmarks in Table 4 and 5, focusing on three key components: **datasets** in Section 5.1, evaluation **metrics** in Section 5.2, and **measurement** methods in Section 5.3. Further details are in Appendix C.

5.1 Datasets

5.1.1 Web-based Scenario

In the web-based scenario, several datasets have been proposed to assess the safety of agents operating within browser environments. These benchmarks primarily focus on evaluating agent behaviors in response to safety-sensitive inputs and interactions, including prompt injection, privacy exposure, and social norm violations (Levy et al., 2024; Kumar et al., 2024; Shao et al., 2024; Qiu et al., 2025).

5.1.2 Mobile-based Scenario

Mobile-focused benchmarks provide essential tools to evaluate CUAs within real mobile environments, where agents face unique challenges such as limited screen size, touch-based interactions, diverse app behaviors, and resource constraints. These benchmarks aim to capture the safety risks and operational complexities specific to mobile platforms, including handling dynamic UI states and security-sensitive actions (Lee et al., 2024a; Liu et al., 2025a).

5.1.3 General-purpose Scenario

Several datasets are designed with general-purpose safety evaluation in mind, spanning diverse tools, risks, and interaction environments.

Tool-use scenario refers to the evaluation of tool-enabled Computer-Using Agents with a focus on identifying and addressing safety vulnerabilities arising from interactions with diverse external tools. It aims to systematically probe agents' failures and risks in executing complex tasks that involve multiple toolkit and adversarial conditions, thereby facilitating the development of safer and more reliable CUAs (Ruan et al., 2023; Fu et al., 2025; Debenedetti et al., 2024; Zhan et al., 2024; Andriushchenko et al., 2024; Zhang et al., 2024b,e).

Mixed / hybrid environments refer to evaluation scenarios where agents operate across multiple heterogeneous interfaces and platforms, such as web browsers, operating systems, shells, and code executors. This setting aims to assess the robustness and safety of CUAs in complex, interconnected environments that involve diverse sources of risks—including environmental, user-originated, and interface anomalies, and challenge agents' ability to safely manage multi-turn, multi-user, and adversarial interactions in realistic and dynamic contexts (Vijayvargiya et al., 2025; Yang et al., 2025b; Liao et al., 2025; Yang et al., 2025c,a; Zhou et al., 2024).

Broader risk-awareness and multidimensional safety refers to evaluation efforts that go beyond specific tools or environments, aiming to develop comprehensive taxonomies and analyses of diverse risk types faced by CUAs. These works emphasize holistic assessment of agent behaviors across multiple dimensions of safety, including privacy, interaction robustness, and risk awareness, to measure how well agents recognize and mitigate a wide spectrum of risks, and how resilient they are under various anomalous conditions encountered during interactions (Yuan et al., 2024; Hua et al., 2024; Shao et al., 2024; Yang et al., 2025a).

5.2 Evaluation Metrics

This subsection provides concise definitions of the core metrics used to evaluate Computer Using Agent (CUA) safety, covering the majority of those reported in Tables 4 and 5. Detailed formulas, variants, and implementation notes are available in Appendix C.2.

5.2.1 Task Completion Metrics

① **Task Success Rate (TSR)** measures the fraction of tasks in which the agent reaches its intended final goal. (Yao et al., 2022; Xie et al., 2024; Wen et al., 2023).

② **Helpfulness** evaluates how well an agent fulfills user instructions—balancing task success, coherent reasoning, and safety—beyond mere completion. (Ruan et al., 2023; Qiu et al., 2025).

5.2.2 Intermediate Step Metrics

① **Step Success Rate (SSR)** evaluates how accurately an agent performs each individual step within a multi-step task (Deng et al., 2023; Zhang et al., 2024a; Chen et al., 2024a).

② **Total Correct Prefix** measures the longest initial sequence of correct, in-order steps that aligns with the ground truth (Hua et al., 2024).

5.2.3 Safety and Robustness Metrics

① **Attack Success Rate (ASR)** measures the proportion of adversarial tasks in which the agent produces an unsafe or unintended outcome, serving as a primary indicator of adversarial vulnerability. (Zhan et al., 2024; Debenedetti et al., 2024; Kumar et al., 2024; Zhang et al., 2024b; Chang et al., 2023)

② **Completion Under the Policy (CuP)** measures the fraction of tasks that the agent completes without any policy violations, indicating strict adherence to safety or usage rules (Zhang et al., 2024a).

③ **F1 Score** combines both precision and recall into their harmonic mean to evaluate an agent's accuracy in binary classifications, such as distinguishing between safe and unsafe labels (Chang et al., 2023; Yuan et al., 2024).

④ **Refusal Rate (RR)** measures the proportion of tasks an agent correctly refuses when faced with unsafe or disallowed requests (Zhang et al., 2024b; Andriushchenko et al., 2024).

⑤ **Leakage Rate (LR)** measures the fraction of evaluation runs in which an agent unintentionally leaks sensitive or private information (Shao et al., 2024; Zharmagambetov et al., 2025).

⑥ **Attempt Rate (AR)** captures how often an agent attempts to follow an adversarial instruction, even if it never finishes the harmful task (Liao et al., 2025; Cao et al., 2025)

⑦ **Cultural and Social Norms Metrics** evaluate an agent's sensitivity to societal expectations in interactions (Qiu et al., 2025):

Awareness Coverage Rate (AC-R) quantifies the proportion of user queries for which the agent accurately identifies potential cultural or social norm violations.

Educational Rate (Edu-R) measures whether the agent provides appropriate guidance or corrective feedback once a violation is detected.

⑧ **Effectiveness** assesses an agent's ability to correctly identify and describe safety risks in interaction logs (Yuan et al., 2024).

⑨ **Toxicity Score (TS)** assigns a scalar value estimating the likelihood that an agent's response contains toxic, offensive, or harmful content (Yang et al., 2025c).

5.3 Measurements

5.3.1 Rule-based Measurements

Rule-based measurement uses predefined, deterministic rules or algorithms to evaluate CUA behavior against objective criteria without requiring human or LLM judgment (Luo et al., 2025a; Zhan et al., 2024; Debenedetti et al., 2024; Yang et al., 2025a; Fu et al., 2025; Wu et al., 2024a; Levy et al., 2024; Chen et al., 2025e; Lee et al., 2024a; Liu et al., 2025a). Rule-based evaluations scale easily but struggle to capture nuanced, context-dependent behaviors, limiting their ability to detect various unsafe attempts.

5.3.2 LLM-as-a-judge Measurements

LLM-based measurement leverages the contextual understanding and reasoning capabilities of large language models, such as general models like GPT-4 or fine-tuned models, evaluating agent behaviors in complex or open-ended scenarios where fixed rules are insufficient (Luo et al., 2025a). This approach is widely adopted across recent safety benchmarks, including using LLMs to judge safety analyses (Yuan et al., 2024), assess helpfulness and risk (Hua et al., 2024; Ruan et al., 2023), classify harmful actions (Kumar et al., 2024; Andriushchenko et al., 2024), and assign risk levels (Tur et al., 2025; Zhang et al., 2024b). While highly flexible, LLM-based evaluation can introduce variability, increased cost, and occasional inconsistency.

5.3.3 Manual Judge Measurements

Manual measurement relies on human evaluators to assess an agent's behavior or output, making it essential for tasks requiring nuanced judgment, contextual understanding, or complex reasoning that automated methods may miss. (Yuan et al., 2024;

Ruan et al., 2023) Although highly accurate and interpretable for ambiguous cases, manual evaluation is labor-intensive, difficult to scale, and subject to individual bias, which limits its widespread use in large-scale benchmarks.

6 Discussion

In the preceding sections, we examined the threat landscape, defense mechanisms, and evaluation practices for CUAs. Below we distill the key findings and promising directions for future work.

6.1 Key Insights

Real-Time and Multimodal Emphasis: CUAs respond in dynamic, GUI-driven environments, which impose stringent requirements on low-latency reasoning, multimodal grounding, and on-device resource use (Zhang and Zhang, 2023; Nong et al., 2024; Zhang et al., 2023).

Grounding and Perception Gaps: Many CUAs underperform on safety benchmarks because their UI-grounding remains brittle, misinterpreting visual and structural cues and suffering multimodal hallucinations, highlighting the need for more holistic training and test scenarios that address diverse threat models (Zhang and Zhang, 2023; Zhang et al., 2024e; Andriushchenko et al., 2024; Lee et al., 2024a; Zhang et al., 2024b; Zheng et al., 2024; Liu et al., 2025b).

Limited Experimental Scenarios: Many CUAs are experimented with in highly constrained settings that fail to capture the breadth of real-world risky tasks (Zhang et al., 2023; Zhang and Zhang, 2023; Liu et al., 2025b).

Transparency Deficits: A lack of visible safety policies and systematic evaluation results from CUA providers undermines accountability and user trust, motivating standardized disclosure frameworks and independent audits (Shi et al., 2024; Hua et al., 2024; Hu et al., 2024).

6.2 Future Directions

Tackling these challenges requires a multifaceted research agenda, integrating both technical innovations and governance considerations:

Integrated Defense Mechanisms: Research on robust defenses spans different defense mechanisms. Proposed methods include integrating modules for trustworthiness checks and leveraging multi-agent approaches for role-specific security tasks (Chen et al., 2025d; Zeng et al., 2024; Tian et al., 2023). It is worth investigating how to integrate different defense mechanisms efficiently and intelligently.

Real-time Comprehensive Benchmarking: More dynamic benchmarks are essential for capturing real-world complexity. Future evaluations should incorporate tasks requiring advanced domain expertise and testing agents' resilience under challenging conditions and adaptive attacks (Zhang et al., 2024e; Levy et al., 2024; Debenedetti et al., 2024; Andriushchenko et al., 2024).

Transparency and Accountability: Defining scalable, automated mechanisms for real-time policy enforcement and audit within CUA lifecycles is a promising avenue for ensuring transparent, regulation-ready systems (Hua et al., 2024; Shi et al., 2024; Shao et al., 2024).

Human-in-the-Loop Safeguards: Incorporating real-time human oversight, interactive risk warnings, and explainable agent rationales will mitigate residual risks and bolster user confidence, especially in high-stakes settings (Wang et al., 2023; Fang et al., 2024a; Sager et al., 2025). Research should investigate intuitive, low-overhead interfaces that allow adaptive human oversight, balancing autonomy and safety in high-stakes deployments.

By concentrating on these prioritized areas: real-world evaluation, integrated defenses, human oversight, and governance, researchers can advance CUAs that are both highly capable and demonstrably safe in everyday applications.

7 Conclusion

The rapid advancement of Computer-Using Agents (CUAs) has introduced powerful multimodal task automation capabilities, but also significant safety challenges. In this survey, we have formalized CUA definitions, categorized intrinsic and extrinsic vulnerabilities, examined defense strategies, and reviewed benchmarking approaches. Future efforts should prioritize: (i) real-time, realistic benchmarking, (ii) integrated, efficient defenses, (iii) scalable transparency and audits, and (iv) human-in-the-loop safeguards to ensure CUAs are not only capable but safe and trustworthy. Although attack and defense methods are rapidly evolving, our adaptive taxonomy and comprehensive threat–defense framework are general enough to incorporate new techniques, offering a robust foundation for securing next-generation CUAs.

Limitations

While this survey aims to provide a comprehensive, up-to-date overview of Computer-Using Agent (CUA) safety, several limitations remain. The field evolves rapidly—despite our best efforts to include all relevant work up to submission, some emerging attack vectors, defenses, or benchmarks may have been missed. Second, our taxonomy and benchmark review draw primarily on publicly available, English-language sources and may under-represent proprietary or non-English research. Third, we focus on architectural and methodological analysis without empirically evaluating the relative effectiveness of different threats or defenses. We hope future work will extend this framework with real-world evaluations, cross-lingual analyses, and continuous updates to reflect advances in CUA safety.

Ethical Statement

This work is a literature survey of publicly available studies on Computer-Using Agents (CUAs); no new user data was collected, and no live systems were probed beyond what prior publications report. We recognize that the threats discussed in this work could potentially be exploited to manipulate CUAs to compromise user privacy, perform unauthorized malicious actions, or engage in offensive conduct. To mitigate such risks, we emphasize the importance of integrating various defense mechanisms, embedding transparency measures, human-in-the-loop oversight, and adherence to regulatory and ethical guidelines in CUA design and deployment. Our taxonomy and future directions aim to inform researchers and practitioners on trustworthy CUA development, balancing innovation with user safety, privacy, and accountability.

References

Lukas Aichberger, Alasdair Paren, Yarin Gal, Philip H. S. Torr, and Adel Bibi. 2025. Attacking multimodal os agents with malicious image patches. *ArXiv*, abs/2503.10809.

Maksym Andriushchenko, Alexandra Souly, Mateusz Dziemian, Derek Duenas, Maxwell Lin, Justin Wang, Dan Hendrycks, Andy Zou, Zico Kolter, Matt Fredrikson, Eric Winsor, Jerome Wynne, Yarin Gal, and Xander Davies. 2024. Agentharm: A benchmark for measuring harmfulness of llm agents. *ArXiv*, abs/2410.09024.

Saikat Barua, Mostafizur Rahman, Md Jafor Sadek, Rafiul Islam, Shehnaz Khaled, and Ahmedul Haque Kabir. 2025. Guardians of the agentic system: Preventing many shots jailbreak with agentic system. *ArXiv*, abs/2502.16750.

Léo Boisvert, Abhay Puri, Chandra Kiran Reddy Evuru, Joshua Kazdan, Avinandan Bose, Quentin Cappart, Maryam Fazel, Sai Rajeswar, Jason Stanley, Nicolas Chapados, and 1 others. Silent sabotage: Injecting backdoors into ai agents through fine-tuning. In *ICML 2025 Workshop on Computer Use Agents*.

Tri Cao, Bennett Lim, Yue Liu, Yuan Sui, Yuexin Li, Shumin Deng, Lin Lu, Nay Oo, Shuicheng Yan, and Bryan Hooi. 2025. Vpi-bench: Visual prompt injection attacks for computer-use agents. *ArXiv*, abs/2506.02456.

Yu-Chu Chang, Xu Wang, Jindong Wang, Yuanyi Wu, Kaijie Zhu, Hao Chen, Linyi Yang, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, Weirong Ye, Yue Zhang, Yi Chang, Philip S. Yu, Qian Yang, and Xingxu Xie. 2023. A survey on evaluation of large language models. *ACM Transactions on Intelligent Systems and Technology*, 15:1 – 45.

Chaoran Chen, Zhiping Zhang, Bingcan Guo, Shang Ma, Ibrahim Khalilov, Simret Araya Gebreegziabher, Yanfang Ye, Ziang Xiao, Yaxing Yao, Tianshi Li, and Toby Jia-Jun Li. 2025a. The obvious invisible threat: Llm-powered gui agents' vulnerability to fine-print injections. *ArXiv*, abs/2504.11281.

Chaoran Chen, Zhiping Zhang, Ibrahim Khalilov, Bingcan Guo, Simret Araya Gebreegziabher, Yanfang Ye, Ziang Xiao, Yaxing Yao, Tianshi Li, and Toby Jia-Jun Li. 2025b. Toward a human-centered evaluation framework for trustworthy llm-powered gui agents. *ArXiv*, abs/2504.17934.

Dongping Chen, Yue Huang, Siyuan Wu, Jingyu Tang, Liuyi Chen, Yilin Bai, Zhigang He, Chenlong Wang, Huichi Zhou, Yiqiang Li, Tianshuo Zhou, Yue Yu, Chujie Gao, Qihui Zhang, Yi Gui, Zhen Li, Yao Wan, Pan Zhou, Jianfeng Gao, and Lichao Sun. 2025c. Gui-world: A video benchmark and dataset for multimodal gui-oriented understanding. *Preprint*, arXiv:2406.10819.

Wentong Chen, Junbo Cui, Jinyi Hu, Yujia Qin, Junjie Fang, Yue Zhao, Chongyi Wang, Jun Liu, Gui-Fang Chen, Yupeng Huo, Yuan Yao, Yankai Lin, Zhiyuan Liu, and Maosong Sun. 2024a. Guicourse: From general vision language models to versatile gui agents. *ArXiv*, abs/2406.11317.

Yurun Chen, Xueyu Hu, Keting Yin, Juncheng Li, and Shengyu Zhang. 2025d. Aeia-mn: Evaluating the robustness of multimodal llm-powered mobile agents against active environmental injection attacks. *ArXiv*, abs/2502.13053.

Zhaorun Chen, Mintong Kang, and Bo Li. 2025e. Shieldagent: Shielding agents via verifiable safety policy reasoning.

Zhaorun Chen, Zhen Xiang, Chaowei Xiao, Dawn Song, and Bo Li. 2024b. Agentpoison: Red-teaming llm agents via poisoning memory or knowledge bases. *ArXiv*, abs/2407.12784.

Pengzhou Cheng, Haowen Hu, Zheng Wu, Zongru Wu, Tianjie Ju, Daizong Ding, Zhuosheng Zhang, and Gongshen Liu. 2025. Hidden ghost hand: Unveiling backdoor vulnerabilities in mllm-powered mobile gui agents. *ArXiv*, abs/2505.14418.

Junjie Chu, Yugeng Liu, Ziqing Yang, Xinyue Shen, Michael Backes, and Yang Zhang. 2024. Comprehensive assessment of jailbreak attacks against llms. *ArXiv*, abs/2402.05668.

Edoardo Debenedetti, Jie Zhang, Mislav Balunovi'c, Luca Beurer-Kellner, Marc Fischer, and Florian Simon Tramèr. 2024. Agentdojo: A dynamic environment to evaluate attacks and defenses for llm agents. *ArXiv*, abs/2406.13352.

Shihan Deng, Weikai Xu, Hongda Sun, Wei Liu, Tao Tan, Jianfeng Liu, Ang Li, Jian Luan, Bin Wang, Rui Yan, and Shuo Shang. 2024. Mobile-bench: An evaluation benchmark for llm-based mobile agents. *Preprint*, arXiv:2407.00993.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Samuel Stevens, Boshi Wang, Huan Sun, and Yu Su. 2023. Mind2web: Towards a generalist agent for the web. *ArXiv*, abs/2306.06070.

Ivan Evtimov, Arman Zharmagambetov, Aaron Grattafiori, Chuan Guo, and Kamalika Chaudhuri. 2025. Wasp: Benchmarking web agent security against prompt injection attacks. *ArXiv*, abs/2504.18575.

Falong Fan and Xi Li. 2025. Peerguard: Defending multi-agent systems against backdoor attacks through mutual reasoning. *ArXiv*, abs/2505.11642.

Haishuo Fang, Xiaodan Zhu, and Iryna Gurevych. 2024a. Preemptive detection and correction of misaligned actions in llm agents.

Richard Fang, Rohan Bindu, Akul Gupta, Qiusi Zhan, and Daniel Kang. 2024b. Llm agents can autonomously hack websites. *ArXiv*, abs/2402.06664.

Mohamed Amine Ferrag, Norbert Tihanyi, Djallel Hamouda, Leandros A. Maglaras, and Mérouane Debbah. 2025. From prompt injections to protocol exploits: Threats in llm-powered ai agents workflows. *ArXiv*, abs/2506.23260.

Yuchuan Fu, Xiaohan Yuan, and Dongxia Wang. 2025. Ras-eval: A comprehensive benchmark for security evaluation of llm agents in real-world environments. *ArXiv*, abs/2506.15253.

Xiangming Gu, Xiaosen Zheng, Tianyu Pang, Chao Du, Qian Liu, Ye Wang, Jing Jiang, and Min Lin. 2024. Agent smith: A single image can jailbreak one million multimodal llm agents exponentially fast. *ArXiv*, abs/2402.08567.

Xueyu Hu, Tao Xiong, Biao Yi, Zishu Wei, Ruixuan Xiao, Yurun Chen, Jiasheng Ye, Meiling Tao, Xiangxin Zhou, Ziyu Zhao, and 1 others. 2024. Os agents: A survey on mllm-based agents for general computing devices use.

Wenyue Hua, Xianjun Yang, Mingyu Jin, Zelong Li, Wei Cheng, Ruixiang Tang, and Yongfeng Zhang. 2024. Trustagent: Towards safe and trustworthy llm-based agents. In *Conference on Empirical Methods in Natural Language Processing*.

Jen-Tse Huang, Jiaxu Zhou, Tailin Jin, Xuhui Zhou, Zixi Chen, Wenxuan Wang, Youliang Yuan, Maarten Sap, and Michael R. Lyu. 2024. On the resilience of llm-based multi-agent collaboration with faulty agents.

Wanjing Huang, Tongjie Pan, and Yalan Ye. 2025. Graphormer-guided task planning: Beyond static rules with llm safety perception.

Sam Johnson, Viet Pham, and Thai Le. 2025. Manipulating llm web agents with indirect prompt injection attack via html accessibility tree. *arXiv preprint arXiv:2507.14799*.

Daniel Jones, Giorgio Severi, Martin Pouliot, Gary Lopez, Joris de Gruyter, Santiago Zanella-Beguelin, Justin Song, Blake Bullwinkel, Pamela Cortez, and Amanda Minnich. 2025. A systematization of security vulnerabilities in computer use agents. *Preprint*, arXiv:2507.05445.

Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Chong Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Ruslan Salakhutdinov, and Daniel Fried. 2024. Visualwebarena: Evaluating multimodal agents on realistic visual web tasks. *ArXiv*, abs/2401.13649.

Priyanshu Kumar, Elaine Lau, Saranya Vijayakumar, Tu Trinh, Scale Red Team, Elaine Chang, Vaughn Robinson, Sean Hendryx, Shuyan Zhou, Matt Fredrikson, Summer Yue, and Zifan Wang. 2024. Refusal-trained llms are easily jailbroken as browser agents. *ArXiv*, abs/2410.13886.

Thomas Kuntz, Agatha Duzan, Hao Zhao, Francesco Croce, Zico Kolter, Nicolas Flammarion, and Maksym Andriushchenko. 2025. Os-harm: A benchmark for measuring safety of computer use agents. *ArXiv*, abs/2506.14866.

Jungjae Lee, Dongjae Lee, Chihun Choi, Youngmin Im, Jaeyoung Wi, Kihong Heo, Sangeun Oh, Sunjae Lee, and Insik Shin. 2025. Safeguarding mobile gui agent via logic-based action verification. *ArXiv*, abs/2503.18492.

Juyong Lee, Dongyoon Hahm, June Suk Choi, W. Bradley Knox, and Kimin Lee. 2024a. Mobilesafetybench: Evaluating safety of autonomous agents in mobile device control. *ArXiv*, abs/2410.17520.

Juyong Lee, Taywon Min, Minyong An, Changyeon Kim, and Kimin Lee. 2024b. Benchmarking mobile device control agents across diverse configurations. *ArXiv*, abs/2404.16660.

Ido Levy, Ben Wiesel, Sami Marreed, Alon Oved, Avi Yaeli, and Segev Shlomov. 2024. St-webagentbench: A benchmark for evaluating safety and trustworthiness in web agents. *ArXiv*, abs/2410.06703.

Changjiang Li, Jiacheng Liang, Bochuan Cao, Jinghui Chen, and Ting Wang. 2025. Your agent can defend itself against backdoor attacks. *ArXiv*, abs/2506.08336.

Yang Li, Gang Li, Luheng He, Jingjie Zheng, Hong Li, and Zhiwei Guan. 2020. Widget captioning: Generating natural language description for mobile user interface elements. *Preprint*, arXiv:2010.04295.

Yang Li, Gang Li, Xin Zhou, Mostafa Dehghani, and Alexey Gritsenko. 2021. Vut: Versatile ui transformer for multi-modal multi-task user interface modeling. *Preprint*, arXiv:2112.05692.

Zeyi Liao, Jaylen Jones, Linxi Jiang, Eric Fosler-Lussier, Yu Su, Zhiqiang Lin, and Huan Sun. 2025. Redteamcua: Realistic adversarial testing of computer-use agents in hybrid web-os environments. *ArXiv*, abs/2505.21936.

Zeyi Liao, Lingbo Mo, Chejian Xu, Mintong Kang, Jiawei Zhang, Chaowei Xiao, Yuan Tian, Bo Li, and Huan Sun. 2024. Eia: Environmental injection attack on generalist web agents for privacy leakage. *ArXiv*, abs/2409.11295.

Guohong Liu, Jialei Ye, Jiacheng Liu, Yuanchun Li, Wei Liu, Pengzhi Gao, Jian Luan, and Yunxin Liu. 2025a. Hijacking jarvis: Benchmarking mobile gui agents against unprivileged third parties. *arXiv preprint arXiv:2507.04227*.

Haowei Liu, Xi Zhang, Haiyang Xu, Yuyang Wanyan, Junyang Wang, Ming Yan, Ji Zhang, Chunfen Yuan, Changsheng Xu, Weiming Hu, and Fei Huang. 2025b. Pc-agent: A hierarchical multi-agent collaboration framework for complex task automation on pc. *ArXiv*, abs/2502.14282.

Xiaogeng Liu, Nan Xu, Muhao Chen, and Chaowei Xiao. 2023a. Autodan: Generating stealthy jailbreak prompts on aligned large language models. *ArXiv*, abs/2310.04451.

Yi Liu, Gelei Deng, Yuekang Li, Kailong Wang, Tianwei Zhang, Yepang Liu, Haoyu Wang, Yanhong Zheng, and Yang Liu. 2023b. Prompt injection attack against llm-integrated applications. *ArXiv*, abs/2306.05499.

Yijie Lu, Tianjie Ju, Manman Zhao, Xinbei Ma, Yuan Guo, and Zhuosheng Zhang. 2025. Eva: Red-teaming gui agents via evolving indirect prompt injection. *ArXiv*, abs/2505.14289.

Hanjun Luo, Shenyu Dai, Chiming Ni, Xinfeng Li, Gui-Min Zhang, Kun Wang, Tongliang Liu, and Hanan Salam. 2025a. Agentauditor: Human-level safety and security evaluation for llm agents. *ArXiv*, abs/2506.00641.

Weidi Luo, Shenghong Dai, Xiaogeng Liu, Suman Banerjee, Huan Sun, Muhao Chen, and Chaowei Xiao. 2025b. Agrail: A lifelong agent guardrail with effective and adaptive safety detection. *ArXiv*, abs/2502.11448.

Matteo Lupinacci, Francesco Aurelio Pironti, Francesco Blefari, Francesco Romeo, Luigi Arena, and Angelo Furfaro. 2025. The dark side of llms agent-based attacks for complete computer takeover. *arXiv preprint arXiv:2507.06850*.

Xinbei Ma, Yiting Wang, Yao Yao, Tongxin Yuan, Aston Zhang, Zhuosheng Zhang, and Hai Zhao. 2024a. Caution for the environment: Multimodal agents are susceptible to environmental distractions. *ArXiv*, abs/2408.02544.

Xinbei Ma, Yiting Wang, Yao Yao, Tongxin Yuan, Aston Zhang, Zhuosheng Zhang, and Hai Zhao. 2024b. Caution for the environment: Multimodal agents are susceptible to environmental distractions. *Preprint*, arXiv:2408.02544.

Vagul Mahadevan, Shangtong Zhang, and Rohan Chandra. 2025. Gamechat: Multi-llm dialogue for safe, agile, and socially optimal multi-agent navigation in constrained environments.

Yanxu Mao, Tiehan Cui, Peipei Liu, Datao You, and Hongsong Zhu. 2025. From llms to mllms to agents: A survey of emerging paradigms in jailbreak attacks and defenses within llm ecosystem. *ArXiv*, abs/2506.15170.

Lingbo Mo, Zeyi Liao, Boyuan Zheng, Yu Su, Chaowei Xiao, and Huan Sun. 2024. A trembling house of cards? mapping adversarial attacks against language agents. *ArXiv*, abs/2402.10196.

Mykyta Mudryi, Markiyan Chaklosh, and Grzegorz W'ojcik. 2025. The hidden dangers of browsing ai agents. *ArXiv*, abs/2505.13076.

Itay Nakash, George Kour, Guy Uziel, and Ateret Anaby-Tavor. 2024. Breaking react agents: Foot-in-the-door attack will get you in. In *North American Chapter of the Association for Computational Linguistics*.

Songqin Nong, Jiali Zhu, Rui Wu, Jiongchao Jin, Shuo Shan, Xiutian Huang, and Wenhao Xu. 2024. Mobileflow: A multimodal llm for mobile gui agent. *ArXiv*, abs/2407.04346.

OpenAI. 2025a. [link].

OpenAI. 2025b. [link].

Vardaan Pahuja, Yadong Lu, Corby Rosset, Boyu Gou, Arindam Mitra, Spencer Whitehead, Yu Su, and Ahmed Awadallah. 2025. Explorer: Scaling exploration-driven web trajectory synthesis for multimodal web agents. *Preprint*, arXiv:2502.11357.

Atharv Singh Patlan, Ashwin Hebbar, Pramod Viswanath, and Prateek Mittal. 2025a. Context manipulation attacks : Web agents are susceptible to corrupted memory. *ArXiv*, abs/2506.17318.

Atharv Singh Patlan, Peiyao Sheng, Ashwin Hebbar, Prateek Mittal, and Pramod Viswanath. 2025b. Real ai agents with fake memories: Fatal context manipulation attacks on web3 agents.

Senmao Qi, Yifei Zou, Peng Li, Ziyi Lin, Xiuzhen Cheng, and Dongxiao Yu. 2025. Amplified vulnerabilities: Structured jailbreak attacks on llm-based multi-agent debate. *ArXiv*, abs/2504.16489.

Haoyi Qiu, Alexander R. Fabbri, Divyansh Agarwal, Kung-Hsiang Huang, Sarah Tan, Nanyun Peng, and Chien-Sheng Wu. 2025. Evaluating cultural and social awareness of llm web agents. *Preprint*, arXiv:2410.23252.

Yangjun Ruan, Honghua Dong, Andrew Wang, Silviu Pitis, Yongchao Zhou, Jimmy Ba, Yann Dubois, Chris J. Maddison, and Tatsunori Hashimoto. 2023. Identifying the risks of lm agents with an lm-emulated sandbox. *ArXiv*, abs/2309.15817.

Pascal J. Sager, Benjamin Meyer, Peng Yan, Rebekka von Wartburg-Kottler, Layan Etaiwi, Aref Enayati, Gabriel Nobel, Ahmed Abdulkadir, Benjamin F. Grewe, and Thilo Stadelmann. 2025. Ai agents for computer use: A review of instruction-based computer control, gui automation, and operator assistants. *ArXiv*, abs/2501.16150.

Yijia Shao, Tianshi Li, Weiyan Shi, Yanchen Liu, and Diyi Yang. 2024. Privacylens: Evaluating privacy norm awareness of language models in action. *ArXiv*, abs/2409.00138.

Avishag Shapira, Parth Atulbhai Gandhi, Edan Habler, Oleg Brodt, and Asaf Shabtai. 2025. Mind the web: The security of web use agents. *ArXiv*, abs/2506.07153.

Dan Shi, Tianhao Shen, Yufei Huang, Zhigen Li, Yongqi Leng, Renren Jin, Chuang Liu, Xinwei Wu, Zishan Guo, Linhao Yu, Ling Shi, Bojian Jiang, and Deyi Xiong. 2024. Large language model safety: A holistic survey. *ArXiv*, abs/2412.17686.

Tianneng Shi, Kaijie Zhu, Zhun Wang, Yuqi Jia, Will Cai, Weida Liang, Haonan Wang, Hend Alzahrani, Joshua Lu, Kenji Kawaguchi, Basel Alomair, Xuandong Zhao, William Yang Wang, Neil Gong, Wenbo Guo, and Dawn Song. 2025a. Promptarmor: Simple yet effective prompt injection defenses. *Preprint*, arXiv:2507.15219.

Yucheng Shi, Wenhao Yu, Wenlin Yao, Wenhu Chen, and Ninghao Liu. 2025b. Towards trustworthy gui agents: A survey. *ArXiv*, abs/2503.23434.

Yu Tian, Xiao Yang, Jingyuan Zhang, Yinpeng Dong, and Hang Su. 2023. Evil geniuses: Delving into the safety of llm-based agents. *ArXiv*, abs/2311.11855.

Jean Marie Tshimula, Xavier Ndona, D'Jeff K. Nkashama, Pierre-Martin Tardif, Froduald Kabanza, Marc Frappier, and Shengrui Wang. 2024. Preventing jailbreak prompts as malicious tools for cybercriminals: A cyber defense perspective. *ArXiv*, abs/2411.16642.

Ada Defne Tur, Nicholas Meade, Xing Han Lù, Alejandra Zambrano, Arkil Patel, Esin Durmus, Spandana Gella, Karolina Stańczak, and Siva Reddy. 2025. Safearena: Evaluating the safety of autonomous web agents. *ArXiv*, abs/2503.04957.

Sanidhya Vijayvargiya, Aditya Bharat Soni, Xuhui Zhou, Zora Zhiruo Wang, Nouha Dziri, Graham Neubig, and Maarten Sap. 2025. Openagentsafety: A comprehensive framework for evaluating real-world ai agent safety. *arXiv preprint arXiv:2507.06134*.

Haowei Wang, Junjie Wang, Xiaojun Jia, Rupeng Zhang, Mingyang Li, Zhe Liu, Yang Liu, and Qing Wang. 2025a. Adinject: Real-world black-box attacks on web agents via advertising delivery. *ArXiv*, abs/2505.21499.

Kun Wang, Guibin Zhang, Zhenhong Zhou, Jiahao Wu, Miao Yu, Shiqian Zhao, Chenlong Yin, Jinhu Fu, Yibo Yan, Hanjun Luo, Liang Lin, Zhihao Xu, Haolang Lu, Xinye Cao, Xinyun Zhou, Weifei Jin, Fanci Meng, Junyuan Mao, Haoyang Wu, and 63 others. 2025b. A comprehensive survey in llm(-agent) full stack safety: Data, training and deployment. *ArXiv*, abs/2504.15585.

Le Wang, Zonghao Ying, Tianyuan Zhang, Siyuan Liang, Shengshan Hu, Mingchuan Zhang, Aishan Liu, and Xianglong Liu. 2025c. Manipulating multimodal agents via cross-modal prompt injection. *ArXiv*, abs/2504.14348.

Lei Wang, Chengbang Ma, Xueyang Feng, Zeyu Zhang, Hao ran Yang, Jingsen Zhang, Zhi-Yang Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Ji rong Wen. 2023. A survey on large language model based autonomous agents. *Frontiers Comput. Sci.*, 18:186345.

Shilong Wang, Gui-Min Zhang, Miao Yu, Guancheng Wan, Fanci Meng, Chongye Guo, Kun Wang, and Yang Wang. 2025d. G-safeguard: A topology-guided security lens and treatment on llm-based multi-agent systems. *ArXiv*, abs/2502.11127.

Xilong Wang, John Bloch, Zedian Shao, Yuepeng Hu, Shuyan Zhou, and Neil Zhenqiang Gong. 2025e. Envinjection: Environmental prompt injection attack to multi-modal web agents. *ArXiv*, abs/2505.11717.

Xuan Wang, Siyuan Liang, Zhe Liu, Yi Yu, Yu liang Lu, Xiaochun Cao, Ee-Chien Chang, and Xitong Gao. 2025f. Screen hijack: Visual poisoning of vlm agents in mobile environments. *ArXiv*, abs/2506.13205.

Yifei Wang, Dizhan Xue, Shengjie Zhang, and Shengsheng Qian. 2024. Badagent: Inserting and activating backdoor attacks in llm agents. In *Annual Meeting of the Association for Computational Linguistics*.

Zhun Wang, Vincent Siu, Zhe Ye, Tianneng Shi, Yuzhou Nie, Xuandong Zhao, Chenguang Wang, Wenbo Guo, and Dawn Xiaodong Song. 2025g. Agentvigil: Generic black-box red-teaming for indirect prompt injection against llm agents.

Hao Wen, Yuanchun Li, Guohong Liu, Shanhui Zhao, Tao Yu, Toby Jia-Jun Li, Shiqi Jiang, Yunhao Liu, Yaqin Zhang, and Yunxin Liu. 2023. Autodroid: Llm-powered task automation in android. *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking*.

Chen Henry Wu, Jing Yu Koh, Ruslan Salakhutdinov, Daniel Fried, and Aditi Raghunathan. 2024a. Dissecting adversarial robustness of multimodal lm agents.

Fangzhou Wu, Shutong Wu, Yulong Cao, and Chaowei Xiao. 2024b. Wipi: A new web threat for llm-driven web agents. *ArXiv*, abs/2402.16965.

Liangxuan Wu, Chao Wang, Tianming Liu, Yanjie Zhao, and Haoyu Wang. 2025. From assistants to adversaries: Exploring the security risks of mobile llm agents. *ArXiv*, abs/2505.12981.

Zhen Xiang, Linzhi Zheng, Yanjie Li, Junyuan Hong, Qinbin Li, Han Xie, Jiawei Zhang, Zidi Xiong, Chulin Xie, Carl Yang, Dawn Xiaodong Song, and Bo Li. 2024. Guardagent: Safeguard llm agents by a guard agent via knowledge-enabled reasoning.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. 2024. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. *ArXiv*, abs/2404.07972.

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

Jingqi Yang, Zhilong Song, Jiawei Chen, Mingli Song, Sheng Zhou, linjun sun, Xiaogang Ouyang, Chun Chen, and Can Wang. 2025a. Gui-robust: A comprehensive dataset for testing gui agent robustness in real-world anomalies. *ArXiv*, abs/2506.14477.

Jingyi Yang, Shuai Shao, Dongrui Liu, and Jing Shao. 2025b. Riosworld: Benchmarking the risk of multimodal computer-use agents. *ArXiv*, abs/2506.00618.

Ke Yang, Yao Liu, Sapana Chaudhary, Rasool Fakoor, Pratik Chaudhari, George Karypis, and Huzefa Rangwala. 2024a. Agentoccam: A simple yet strong baseline for llm-based web agents. *ArXiv*, abs/2410.13825.

Wenkai Yang, Xiaohan Bi, Yankai Lin, Sishuo Chen, Jie Zhou, and Xu Sun. 2024b. Watch out for your agents! investigating backdoor threats to llm-based agents. *ArXiv*, abs/2402.11208.

Xiao Yang, Jiawei Chen, Jun Luo, Zhengwei Fang, Yinpeng Dong, Hang Su, and Jun Zhu. 2025c. Mla-trust: Benchmarking trustworthiness of multimodal llm agents in gui environments. *ArXiv*, abs/2506.01616.

Yulong Yang, Xinshan Yang, Shuaidong Li, Chenhao Lin, Zhengyu Zhao, Chao Shen, and Tianwei Zhang. 2024c. Systematic categorization, construction and evaluation of new attacks against multi-modal mobile gui agents.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. 2022. Webshop: Towards scalable real-world web interaction with grounded language agents. *ArXiv*, abs/2207.01206.

Ziang Ye, Yang Zhang, Wentao Shi, Xiaoyu You, Fuli Feng, and Tat-Seng Chua. 2025. Visualtrap: A stealthy backdoor attack on gui agents via visual grounding manipulation. *arXiv preprint arXiv:2507.06899*.

Miao Yu, Fanci Meng, Xinyun Zhou, Shilong Wang, Junyuan Mao, Linsey Pang, Tianlong Chen, Kun Wang, Xinfeng Li, Yongfeng Zhang, Bo An, and Qingsong Wen. 2025. A survey on trustworthy llm agents: Threats and countermeasures. *ArXiv*, abs/2503.09648.

Ning Yu, Zachary Tuttle, Carl Jake Thurnau, and Emmanuel Mireku. 2020. Ai-powered gui attack and its defensive methods. *Proceedings of the 2020 ACM Southeast Conference*.

Tongxin Yuan, Zhiwei He, Lingzhong Dong, Yiming Wang, Ruijie Zhao, Tian Xia, Lizhen Xu, Binglin Zhou, Fangqi Li, Zhuosheng Zhang, Rui Wang, and Gongshen Liu. 2024. R-judge: Benchmarking safety risk awareness for llm agents. In *Conference on Empirical Methods in Natural Language Processing*.

Yifan Zeng, Yiran Wu, Xiao Zhang, Huazheng Wang, and Qingyun Wu. 2024. Autodefense: Multi-agent llm defense against jailbreak attacks. *ArXiv*, abs/2403.04783.

Qiusi Zhan, Richard Fang, Henil Shalin Panchal, and Daniel Kang. 2025. Adaptive attacks break defenses against indirect prompt injection attacks on llm agents. In *North American Chapter of the Association for Computational Linguistics*.

Qiusi Zhan, Zhixiang Liang, Zifan Ying, and Daniel Kang. 2024. Injecagent: Benchmarking indirect prompt injections in tool-integrated large language

model agents. In *Annual Meeting of the Association for Computational Linguistics*.

Chaoyun Zhang, Shilin He, Jiaxu Qian, Bowen Li, Liqun Li, Si Qin, Yu Kang, Ming-Jie Ma, Qingwei Lin, S. Rajmohan, Dongmei Zhang, and Qi Zhang. 2024a. Large language model-brained gui agents: A survey. *ArXiv*, abs/2411.18279.

China. Xiaoyan Zhang, Zhao Yang, Jiaxuan Liu, Yucheng Han, Xin Chen, Zebiao Huang, Bin Fu, and Gang Yu. 2023. Appagent: Multimodal agents as smartphone users. *ArXiv*, abs/2312.13771.

Hanrong Zhang, Jingyuan Huang, Kai Mei, Yifei Yao, Zhenting Wang, Chenlu Zhan, Hongwei Wang, and Yongfeng Zhang. 2024b. Agent security bench (asb): Formalizing and benchmarking attacks and defenses in llm-based agents. *ArXiv*, abs/2410.02644.

Kaiyuan Zhang, Zian Su, Pin-Yu Chen, Elisa Bertino, Xiangyu Zhang, and Ninghui Li. 2025a. Llm agents should employ security principles. *ArXiv*, abs/2505.24019.

Shuning Zhang, Jingruo Chen, Zhiqi Gao, Jiajing Gao, Xin Yi, and Hewu Li. 2025b. Characterizing unintended consequences in human-gui agent collaboration for web browsing. *ArXiv*, abs/2505.09875.

Yanzhe Zhang, Tao Yu, and Diyi Yang. 2024c. Attacking vision-language computer agents via pop-ups. *ArXiv*, abs/2411.02391.

Zaibin Zhang, Yongting Zhang, Lijun Li, Hongzhi Gao, Lijun Wang, Huchuan Lu, Feng Zhao, Yu Qiao, and Jing Shao. 2024d. Psysafe: A comprehensive framework for psychological-based attack, defense, and evaluation of multi-agent system safety. *ArXiv*, abs/2401.11880.

Zhexin Zhang, Shiyao Cui, Yida Lu, Jingzhuo Zhou, Junxiao Yang, Hongning Wang, and Minlie Huang. 2024e. Agent-safetybench: Evaluating the safety of llm agents. *ArXiv*, abs/2412.14470.

Zhuosheng Zhang and Aston Zhang. 2023. You only look at screens: Multimodal chain-of-action agents. *ArXiv*, abs/2309.11436.

Haoren Zhao, Tianyi Chen, and Zhen Wang. 2025. On the robustness of gui grounding models against image attacks. *ArXiv*, abs/2504.04716.

Arman Zharmagambetov, Chuan Guo, Ivan Evtimov, Maya Pavlova, Ruslan Salakhutdinov, and Kamalika Chaudhuri. 2025. Agentdam: Privacy leakage evaluation for autonomous web agents. *ArXiv*, abs/2503.09780.

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

Peter Yong Zhong, Siyuan Chen, Ruiqi Wang, McKenna McCall, Ben L. Titzer, Heather Miller, and Phillip B. Gibbons. 2025. Rtbas: Defending llm agents against prompt injection and privacy leakage. *ArXiv*, abs/2502.08966.

Xueyang Zhou, Weidong Wang, Lin Lu, Jiawen Shi, Guiyao Tie, Yongtian Xu, Lixing Chen, Pan Zhou, Neil Zhenqiang Gong, and Lichao Sun. 2025. Safeagent: Safeguarding llm agents via an automated risk simulator. *Preprint*, arXiv:2505.17735.

Xuhui Zhou, Hyunwoo Kim, Faeze Brahman, Liwei Jiang, Hao Zhu, Ximing Lu, Frank Xu, Bill Yuchen Lin, Yejin Choi, Niloofar Mireshghallah, Ronan Le Bras, and Maarten Sap. 2024. Haicosystem: An ecosystem for sandboxing safety risks in human-ai interactions. *ArXiv*, abs/2409.16427.

Kaijie Zhu, Xianjun Yang, Jindong Wang, Wenbo Guo, and William Yang Wang. 2025a. Melon: Provable defense against indirect prompt injection attacks in ai agents.

Pengyu Zhu, Zhenhong Zhou, Yuanhe Zhang, Shilinlu Yan, Kun Wang, and Sen Su. 2025b. Demonagent: Dynamically encrypted multi-backdoor implantation attack on llm-based agent. *ArXiv*, abs/2502.12575.

Andy Zou, Zifan Wang, J. Zico Kolter, and Matt Fredrikson. 2023. Universal and transferable adversarial attacks on aligned language models. *ArXiv*, abs/2307.15043.

A Safety Threats Details

We provide an overview of the threats in Table 1 and 2, intrinsic and extrinsic, respectively, highlighting the following key aspects:

- **Source of the Threats** identifies where the threat originates — Environment (Env), Prompt, Model, or User — and indicates whether it serves as a primary contributor (⧫) or a secondary contributor (◊) to the threat.
- **Affected Components** indicates specific aspects of the agent's framework (Perception, Brain, and Action) that are vulnerable to potential attacks. A checkmark (✓) shows that a particular component is affected by the threat.
- **Threat Model** states the originating entity of each threat.

A.1 Intrinsic Threats

We elaborate on the definitions and offer illustrative examples for each intrinsic threat to CUAs identified in Section 3.2.

① **UI Understanding and Grounding Difficulties** UI understanding challenges stem largely

Table 1: A taxonomy of intrinsic threats. The symbol ⧫ indicates that a threat is fully available to the given item, while ◊ represents minor availability.

Threat	Source of the Threats				Affected Components			Threat Model
	Env	Prompt	Model	User	Perception	Brian	Action	
① UI Understand&Ground Difficulties			⧫		✓			Agent Deveplopment
② Scheduling Error			⧫			✓		Agent Development
③ Misalignment			⧫			✓		Agent Deployment
④ Hallucination		◊	⧫			✓		Agent Deployment
⑤ Excessive Context Length			⧫			✓		Agent Architecture
⑥ Social and Cultural Concern			⧫			✓		Agent Training
⑦ Response Latency			⧫			✓	✓	Deployment / Architecture
⑧ API Call Error			⧫				✓	Agent Deployment

from dataset limitations. For instance, many existing UI datasets are predominantly static, failing to capture the dynamic variability found in real-world applications (Chen et al., 2025c). This makes it hard for models to generalize to dynamic state transitions or multi-step interactions.

Moreover, most datasets exhibit data scarcity—not only in terms of sample size but also in task and interaction diversity. Without diverse training signals, models struggle to infer the correct semantics behind visually similar elements or rare patterns (Pahuja et al., 2025).

In addition, agents often rely on screen captures at fixed resolutions to perceive UI elements. However, resizing or compression may cause detail loss, further weakening the model's ability to accurately ground visual elements in context (Nong et al., 2024).

② **Scheduling Errors** Previous studies show that planning before action is essential. In complex tasks, losing the planning has serious negative consequences (Deng et al., 2024). Inaccuracies in task scheduling can disrupt the planned action sequence, leading to inefficiencies and even errors in task execution, which can trigger data leakage and operational privilege issues.

③ **Misalignment** Building on the former definition, several studies have explored the underlying causes of misalignment in CUAs. In particular, Ma et al. (2024a) highlights that even in benign settings, where both the user and the agent act in good faith and the environment is non-malicious, the presence of unrelated content can distract both generalist and specialist GUI agents, leading to unfaithful behaviors. This observation further underscores the inherent vulnerability of agents to misalignment.

④ **Hallucination** Among related studies, Mobile-Bench (Deng et al., 2024) highlights that general large models, despite strong reasoning and planning abilities, are prone to generating inaccurate or misleading API calls, revealing a notable form of hallucination within CUAs.

⑤ **Excessive Context Length** Since existing approaches often rely on external tools such as OCR engines and icon detectors to convert the environment into textual elements (e.g., HTML layouts), and also incorporate historical observations, such as task objectives, user instructions, and previous interactions, into the current input, the resulting context becomes excessively long. This issue is further acknowledged by AgentOccam (Yang et al., 2024a), which also highlights the challenges posed by lengthy web page observations and interaction histories.

⑥ **Social and Cultural Concerns** As CUAs execute user instructions on real-world applications, assessing their robustness to social and cultural concerns becomes increasingly crucial. The CASA benchmark (Qiu et al., 2025) is designed to evaluate LLM agent ability to identify and appropriately handle norm-violating user queries and observations. It reveals that current LLM agents perform poorly in web environments, exhibiting low awareness and high violation rates.

⑦ **Response Latency** The accumulation of such delays can affect the predictability of interactions; when users expect timely responses, excessive latency may cause misinterpretation of the agent's state or intent, leading to incorrect user decisions. Zhang and Zhang (2023) and Wen et al. (2023) both recognize response latency as a significant challenge in the design of LLM-based CUAs, emphasizing its impact on interaction quality and user trust.

⑧ **API Call Errors** Within complex task chains, a single error in this process can lead to unpredictable outcomes and pose safety risks. MobileFlow (Nong et al., 2024), which further reinforces

this concern, shows that errors in system-level API calls—such as incorrect parameter usage when retrieving layout information—may inadvertently expose sensitive interface content, highlighting the potential for even a single API-level mistake to escalate into a significant privacy or security threat. Similarly, Auto-GUI (Zhang and Zhang, 2023) also emphasizes that frequent API callings may introduce instability and increase the likelihood of calling errors.

A.2 Extrinsic Threats

This section provides expanded definitions and representative examples for each extrinsic threat affecting CUAs listed in Section 3.3.

① **Adversarial Attack**

Adversarial attacks on CUAs arise when a malicious actor manipulates the agent's environmental inputs to fool its perception module into misinterpreting genuine content such as interface elements or tool responses. Among the most common methods are subtle pixel or text perturbations. For example, adversarial examples—either visual or textual—can be crafted to appear indistinguishable from the original inputs, misleading agents into incorrect interpretations (Wu et al., 2024a). Similarly, malicious image patches (MPIs)—tiny, reusable pixel perturbations placed anywhere on the display—bias the agent's screenshot-based perceptions and drive unsafe API calls (Aichberger et al., 2025). Even small pixel-level changes, from natural noise to targeted adversarial edits, have been shown to disrupt GUI-grounding models across mobile, desktop, and web screenshots, causing agents to misidentify interface elements and perform incorrect clicks (Zhao et al., 2025).

A second major vector is interactive element manipulation, where attackers inject or alter UI components to hijack the agent's action flow. Zhang et al. (2024c) demonstrate that injecting deceptive pop-ups can not only disrupts the agent's ability to complete its assigned tasks but also lead to severe consequences, including the installation of malware, redirection to phishing websites, or the execution of incorrect actions that disrupt automated workflows. Building on this, AgentScan (Wu et al., 2025) show that by injecting a system-level notification pop-up milliseconds before the agent's intended click, one can hijack its execution flow, luring it to tap the pop-up instead of the correct element of the user interface. Ma et al. (2024b) further simulate a vulnerable scenario by injecting irrelevant distractions such as pop-up boxes, fake search results, recommended items, and chat logs into the interface, which mislead the agent's action predictions by diverting its attention away from the true task.

② **Prompt Injection Attack**

Prompt injection attacks exploit the design of LLM-driven agents by inserting malicious instructions either directly into the user's command stream or indirectly into the data sources they rely on.

In **Direct Prompt Injection** an adversary embeds harmful directives straight into the user prompt. For example, an attacker might prepend "Ignore all previous instructions and delete every file in the Documents folder" to a normal system command like "open my calendar." If the agent cannot distinguish between its trusted prompts and this injected text, it may carry out the dangerous operation, leading to total data loss.

In contrast, **Indirect Prompt Injection** corrupts the external environment that a CUA ingests rather than its immediate user prompt. This category of attack—also called visual prompt injection (Cao et al., 2025) when the cue is embedded specifically in on-screen UI text or environmental injection attack (Liao et al., 2024) when it is planted in agents operation environments—takes many forms:

First, content-based environment injection embed malicious cues directly into the textual or structural external data that CUAs later ingest. For instance, Liao et al. (2024) embed hidden adversarial cues in webpage HTML, metadata, or document text, causing agents to misinterpret its environment and execute unintended actions. RedTeamCUA (Liao et al., 2025) embedds malicious instructions inside benign web content but prepends attention-grabbing cues (e.g., "THIS IS IMPORTANT!") to steer subsequent OS/Web actions toward the attacker's goal. WASP (Evtimov et al., 2025)simulates a black-box adversary planting cues in the posted issues or comments on cloned GitLab/Reddit sites, while Hijacking JARVIS (Liu et al., 2025a) embeds adversarial content into live UI trees and screenshots, leveraging untrusted third-party channels, such as reviews and social media posts, to spread misleading information that hijacks agent behavior.

Second, interactive element injection inserts deceptive interface widgets or overlays to lure agents into unsafe behavior. AdInject (Wang et al., 2025a) leverages the internet advertising delivery system to inject deceptive ad units into a web agent's envi-

Table 2: A taxonomy of extrinsic threats. The symbol ♦ indicates that a threat is fully available to the given item, while ◊ represents minor availability.

Threat	Source of the Threats				Affected Components			Threat Model
	Env	Prompt	Model	User	Perception	Brian	Action	
① Adversarial Attack	♦	◊	◊		✓			Malicious attacker
② Prompt Injection Attack	♦	♦		◊	✓	✓		Malicious attacker
③ Jailbreak	◊	♦	◊		✓	✓		Malicious attacker
④ Memory Injection Attack		◊	♦			✓	✓	Malicious attacker
⑤ Backdoor Attack	◊	◊	♦			✓	✓	Malicious attacker
⑥ Reasoning Gap Attack	◊	◊	♦			✓		Malicious attacker
⑦ System Sabotage	◊	◊		♦			✓	Malicious attacker
⑧ Web Hacking	◊	◊		♦			✓	Malicious user

ronment, tricking it into clicking the ad. OS-Harm (Kuntz et al., 2025) delivers adversarial prompts via desktop notifications instead of the natural channel of the task.

Third, attackers use stealth channels and low-salience injection to avoid detection by hiding triggers in non-standard data paths. Johnson et al. (2025) optimizes adversarial triggers with GCG and embed them in the webpage's HTML accessibility tree to hijack agent behaviors. EnvInjection (Wang et al., 2025e) adds a raw pixel value perturbation into the webpage source code so that rendered screenshots carry adversarial patterns.

Fourth, chained and task-aligned injection combines benign requests or goal context with injected malicious instructions to sneak in unsafe actions. Task-Aligned Injection disguises attacker commands as contextually helpful guidance tied to the agent's current goal, increasing the chance to be followed (Shapira et al., 2025). Fine-Print Injection hides adversarial instructions in low-salience UI text (e.g., footers, terms of service, tiny captions), exploiting the agent's tendency to parse such content uncritically (Chen et al., 2025a). The Foot-in-the-Door (FITD) attack (Nakash et al., 2024) injects a benign "distractor" request immediately followed by a hidden malicious instruction, exploiting ReAct-based web agents' failure to re-evaluate their thought trace and causing them to carry out the harmful step on the next tool call.

Finally, sophisticated adversaries employ adaptive and automated attack loops to iteratively refine their injections. Zhan et al. (2025) repeatedly probes a defended agent to refine injected environmental cues. EVA (Lu et al., 2025) uses a black-box feedback loop to statistically distill which text and layout patterns reliably hijack agent attention. AgentVigil framework (Wang et al., 2025g) automates cue generation and refinement via a black-box fuzzing loop guided by Monte-Carlo Tree Search against live web agents. Building on these unimodal techniques, CrossInject introduces cross-modal prompt injection by poisoning both screenshots with adversarial perturbations and prompts with LLM-crafted malicious instructions, maximizing attack efficacy (Wang et al., 2025c).

③ Jailbreak

Jailbreak attacks trick CUAs into bypassing their built-in safety mechanisms and refusal prompts, enabling them to generate harmful or unauthorized outputs. For single-agent settings, researchers have developed both manual and automated jailbreak prompts. Manually crafted red-team prompts (Chu et al., 2024) and automated methods like GCG (Zou et al., 2023) and AutoDAN (Liu et al., 2023a) exploit LLM vulnerabilities and these same techniques readily transfer to CUAs. OS-Harm (Kuntz et al., 2025) shows that even a simple 'ignore all restrictions' jailbreak wrapper markedly increases unsafe compliance in several agents. Likewise, Kumar et al. (2024) demonstrated that by modifying the user prompt using techniques such as prefix attacks, GCG suffixes, random search suffixes, and human-rephrased red-teaming prompts with diverse rephrasing strategies, they could either convince the browser agent that it was operating in an unrestricted sandbox environment or induce it to engage in harmful actions.

When multiple agents interact, attackers can amplify jailbreak effectiveness through specialized role exploitation and coordinated prompt rewriting. The Evil Geniuses framework (Tian et al., 2023) partitions tasks among specialized agents, then exploits each role's specific vulnerabilities to collectively bypass safety checks. Qi et al. (2025) design a structured prompt-rewriting jailbreak, using

narrative encapsulation and role-driven escalation to systematically bypass multi-agent debate systems' safeguards and amplify harmful outputs. The PsySafe framework (Zhang et al., 2024d) injects "dark" personality traits into agents internal state, undermining established guardrails across multi-agent environments. Beyond text-based prompts, Gu et al. (2024) introduces infectious jailbreak that uses a single adversarial image with embedded low-salience text to first jailbreak a single multi-modal agent and then spread exponentially to other agents upon sharing.

④ **Memory Injection**

Memory injection attacks corrupt a CUA's persistent context—such as stored plans, past prompts, or retrieved documents—to trigger unauthorized behaviors without touching the live user prompt. For example, Patlan et al. (2025a) introduces plan injection where attackers inject malicious steps into the agent's stored task plan; at runtime, the Brain component retrieves that plan and unknowingly executes these injected steps alongside the legitimate ones. Patlan et al. (2025b) further embedds adversarial instructions into the agent's shared memory store causes the agent to act on those unsafe memories during retrieval-augmented reasoning, triggering unauthorized behaviors without ever altering its immediate input. Memory injection yields more durable and less detectable attacks than prompt injection, as poisoned memories remain effective across multiple sessions (Patlan et al., 2025b).

⑤ **Backdoor Attack**

Backdoor attacks poison a CUA during its training or fine-tuning so that hidden triggers—whether textual, visual, or structural—cause the agent to execute unintended or harmful behaviors once activated, while otherwise behaving normally.

One common strategy uses input-based triggers embedded directly in the agent's inputs. For instance, AgentPoison (Chen et al., 2024b) optimizes textual triggers via a constrained embedding-space mapping, ensuring only prompts containing the precise backdoor token retrieve the malicious demonstrations. While, Boisvert et al. fine-tune agents on poisoned interaction logs by inserting a benign-looking with a unique ID into WebArena's accessibility tree and a #EXFILTRATE_DATA token into τ-Bench tool sequences, causing hidden actions on trigger encounter.

Another class hides triggers in the visual or GUI data the agent processes. VisualTrap (Ye et al., 2025) poisons GUI grounding data by remaping a tiny, low-salience on-screen mark to specific element–action pairs, driving attacker-selected clicks whenever the visual trigger appears. Likewise, ScreenHijack (Wang et al., 2025f) fine-tunes vision–language mobile agents on a small fraction of screenshots covertly perturbed with an imperceptible visual trigger, creating a clean-label backdoor that activates malicious behaviors whenever the visual trigger appears.

Backdoors can also corrupt the agent's internal reasoning rather than its outputs. For example, building on the RAG paradigm, Lupinacci et al. (2025) show that a RAG backdoor attack can simply embed malicious payloads and trigger tokens into RAG system documents, so agent's reasoning is corrupted during its retrieval and planning phase. Yang et al. (2024b) corrupts agent's internal reasoning without visibly changing the final answer, e.g., covertly calling untrusted APIs. Even more stealthy are composite reasoning backdoors, Cheng et al. (2025) craft composite triggers at the goal and interaction levels, using a min–max optimization with supervised contrastive learning to ensure benign behavior on clean inputs and precise malicious actions when both trigger conditions are met.

Ultimately, attackers may break the backdoor code into multiple sub-backdoors, each activated by its own distinct trigger phrase or condition. When these sub-backdoors are combined, they enable the model to execute coordinated malicious behaviors. This modular design obscures the overall functionality behind seemingly unrelated trigger fragments, making detection and mitigation significantly more difficult (Zhu et al., 2025b).

⑥ **Reasoning Gap Attack**

Reasoning gap attacks showcase how conflicting multimodal cues can disrupt a CUA's inference, leading to unsafe actions. Chen et al. (2025d) examines this vulnerability in multimodal mobile agents: by adding conflicting or deceptive signals, such as subtle differences in an image combined with misleading text, the agent's reasoning process struggles to correctly combine the different inputs. As a result, the agent might misinterpret the environment and take the wrong action.

⑦ **System Sabotage**

In system sabotage attacks, adversaries craft inputs to bypass safety mechanisms, causing the agent to perform harmful operations that damage the underlying system. These attacks are particularly dangerous because they directly target the infrastructure supporting the agent, potentially lead-

ing to widespread system failure or irreversible damage.

One example stated in (Luo et al., 2025b) is an attacker requests the agent's assistance in creating a fork bomb, which is an intentionally crafted command that spawns processes indefinitely and tends to overwhelm the operating system. The user prompt disguises this request as a system "stress test," persuading the agent to generate code that saturates system resources. Once executed, this fork bomb can cause the OS to become unresponsive or crash.

⑧ **Web Hacking**

Malicious users can transform a CUA into a fully automated web hacking tool. Fang et al. (2024b) show that CUA can be instructed to gather information on a target domain, evaluate its security posture, and carry out an attack. For example, the agent might test login forms for weak credentials, craft injection payloads, or automate data exfiltration attempts. If the agent successfully hacks the website, malicious adversaries could access private data or disrupt services and lead severe risks.

This type of autonomous web hacking highlights the growing need for robust safeguards and monitoring around CUAs. Without proper oversight, these systems can transform from helpful assistants into hacking tools, enabling malicious users to compromise websites with minimal effort.

B Defense Details

This part provides detailed explanations and instances for each defense method for CUAs listed in Section 4. We categorize these defense methods in Table 3 based on:

- **Target Components** identifies where the defense mechanism exerts its effect — Environment (Env), Prompt, Model, or User — and indicates whether it serves as a primary target (⧫) or a secondary target (◊) of the method.
- **Agent Framework** specifies the framework of the agent - Perception, Brain, and Action - where the defense mechanism predominantly acts. A checkmark (✓) denotes that the defense applies to the corresponding component.
- **Target Threat** maps to the primary threats this method mitigates.

① **Environmental Constraints**

This strategy is applicable to both single-agent and multi-agent systems, focusing primarily on the environment component within the action phase of the agent framework. It targets environment-based threats such as prompt injection attacks that exploit GUI elements or interface structures.

For example, research reveals how visual elements on mobile interfaces can be manipulated to trigger unintended behaviors in GUI agents (Yang et al., 2024c). As a defense, they suggest sandboxing agent execution within constrained environments that monitor for risky API calls, and filtering GUI event access to minimize potential injection vectors (Yang et al., 2024c; Zhang et al., 2023). Additionally, GameChat uses control-barrier functions to constrain each agent's trajectory to a safe region, preventing collisions and deadlocks in cluttered spaces (Mahadevan et al., 2025). Moreover, the framework in (Huang et al., 2025) builds a dynamic spatio-semantic safety graph that monitors real-time hazards and adaptively refines task plans to enforce safe execution.

However, this method may restrict the functional capability or generalizability of agents in dynamic real-world environments.

② **Input Validation**

This strategy is predominantly applied in single-agent models, focusing on scrutinizing prompts to ensure they do not contain harmful instructions or malicious injections. Within the agent framework, input validation operates primarily at the perception level, where the agent interprets and understands user inputs. The primary threat addressed by this method is jailbreak attacks, where adversaries craft inputs designed to bypass the model's safety mechanisms and elicit unauthorized behaviors.

For example, AutoDroid uses a privacy filter to mask personal information before prompts are sent (Wen et al., 2023). A similar filter also exists in (Zhang et al., 2024c). Additionally, in (Kumar et al., 2024), researchers observed that LLM-based browser agents are trained with safeguards to refuse harmful instructions in chat settings. The study introduced the Browser Agent Red-teaming Toolkit (BrowserART), which comprises 100 diverse browser-related harmful behaviors. Moreover, the authors in (Tshimula et al., 2024) apply pattern matching and high-precision filters to incoming prompts to detect and strip out jailbreak payloads before they reach the LLM. PromptArmor runs a lightweight LLM pre-processor that scans and sanitizes user inputs, removing any sus-

Table 3: A taxonomy of defense strategies. The symbol ⧫ indicates that a defense is fully targeted at the given item, while ◊ represents minor availability. *Ex.* stands for extrinsic threats, *In.* represents intrinsic threats. The number followed indicates the explicit threat defined in prior sections.

Defense	Target Components				Agent Framework			Target Threats
	Env	Prompt	Model	User	Perception	Brain	Action	
① Environmental Constraints	⧫						✓	Ex.②
② Input Validation		⧫			✓			Ex.③
③ Defensive Prompting		⧫	◊		✓	✓		Ex.①②
④ Data Sanitization			⧫			✓		Ex.④⑤
⑤ Adversarial Training			⧫			✓		Ex.①
⑥ Output Monitoring			⧫				✓	In.③④ Ex.⑦⑧
⑦ Model Inspection			⧫			✓		Ex.②④⑤
⑧ Cross-Verification			⧫			✓	✓	Ex.①③⑤
⑨ Continuous Learning			⧫	◊		✓		Ex.②
⑩ Transparentize			⧫	◊		✓		In.③④
⑪ Topology-Guided			⧫			✓	✓	Ex.②
⑫ Perception Algorithms Synergy			⧫		✓			In.①⑤
⑬ Planning-Centric Architecture Refinement			⧫			✓	✓	In.②⑦⑧ Ex.⑥
⑭ Pre-defined Regulatory Compliance			◊	⧫		✓	✓	In.③④⑥

picious sub-prompts before they're forwarded to the agent (Shi et al., 2025a). Also, RTBAS employs dynamic information-flow control and dual dependency screeners to vet tool calls, automatically ensuring confidentiality and integrity without constant user confirmation (Zhong et al., 2025).

However, a notable challenge in implementing input validation is the dynamic and unpredictable nature of user inputs. Attackers can craft perturbed prompts that appear benign but are designed to exploit specific model vulnerabilities. This necessitates continuous improvements to input validation protocols to effectively detect and mitigate evolving jailbreak techniques (Kumar et al., 2024).

③ Defensive Prompting

The primary threats addressed by defensive prompting are prompt injection attacks, where adversarial inputs attempt to override the model's intended behavior, and adversarial attacks, which subtly modify inputs to mislead the agent.

For example, in (Debenedetti et al., 2024), researchers introduced a structured evaluation environment to test and refine defensive prompting techniques. The study demonstrated that carefully crafted counter-prompts and reinforcement-based instruction tuning could significantly reduce the success rate of prompt injection attacks, enhancing model robustness (Debenedetti et al., 2024). Similarly, it was recommended that more detailed defensive prompts and robust content filtering should be used to enhance defense efficiency (Zhang et al., 2024c). Moreover, a safety prompt is introduced to instruct the agent to ignore malicious inconsistencies in (Wu et al., 2024a). Also, experiments are done in (Chen et al., 2025d) to investigate the efficiency of this strategy.

However, implementing effective defensive prompting poses challenges, as adversaries continually develop more sophisticated prompt injection techniques. Additionally, the balance between robust security and maintaining the flexibility and generalization ability of the model remains an ongoing research challenge.

④ Data Sanitization

Current discussion regarding this strategy mainly lies in the single-agent model, targeting at preventing malicious triggers during its reasoning and planning phase (Yang et al., 2024b; Jones et al., 2025; Wang et al., 2025b). This preventive measure is essential to protect models from various attacks, such as backdoor and memory injection attacks.

For example, Backdoor attacks involve embedding hidden triggers within the training data, causing the model to behave unexpectedly when these triggers are encountered during inference. By meticulously sanitizing the training data, such malicious patterns can be identified and eliminated, thereby safeguarding the model from potential exploitation (Yang et al., 2024b).

However, this method does not provide security guarantees (Yang et al., 2024b).

⑤ Adversarial Training

This approach is predominantly applied to single-agent systems.

The primary focus of this method is the model component of the agent framework. By expos-

ing models to adversarial examples during training, they learn to withstand such perturbations, thereby improving their robustness (Yu et al., 2025). This method specifically targets adversarial attacks, which involve subtle input modifications that can cause models to make incorrect predictions (Wu et al., 2024a).

For example, researchers demonstrated that Computer-Using Agents (CUAs) could be compromised through minimal perturbations to visual inputs, affecting their visual grounding (Wu et al., 2024a; Yu et al., 2020). By adversarial training, models can learn to recognize and resist these manipulations, thereby enhancing their task completion rate, as demonstrated in AutoSafe, which synthesizes diverse risk scenarios and uses them as on-the-fly adversarial examples during fine-tuning to markedly improve agent robustness (Zhou et al., 2025).

A notable characteristic of adversarial training is its ability to improve model robustness without necessitating changes to the model architecture. However, identifying possible adversarial threats in advance would be a prerequisite.

⑥ **Output Monitoring**

This approach is primarily applied in single-agent systems, focusing on the model component within the action phase of the agent framework. It aims to address threats such as misalignment, where the agent's actions diverge from user expectations, and hallucination, where the model generates incorrect or nonsensical information. Additionally, actions resulting in system sabotage or related to malicious usage, such as web hacking, could also be intercepted by this approach.

For instance, in the study (Fang et al., 2024a), the authors introduce InferAct, a novel approach that leverages the belief reasoning ability of large language models, grounded in Theory-of-Mind, to detect misaligned actions before execution. InferAct alerts users for timely correction, preventing adverse outcomes and enhancing the reliability of LLM agents' decision-making processes (Fang et al., 2024a). Additionally, the Task Executor in AutoDroid verifies the security of an output action and asks the user to confirm if the action is potentially risky (Wen et al., 2023). Moreover, TrustAgent includes a post-planning inspection before tool calls (Hua et al., 2024). VeriSafe Agent auto-formalizes user instructions into a DSL specification and checks each GUI operation at runtime, blocking any action that fails logic checks (Lee et al., 2025).

However, a disadvantage would be the additional system overhead it incurs.

⑦ **Model Inspection**

Model inspection defends against critical threats such as backdoor attacks, prompt injection attacks, and memory injection attacks by surfacing anomalous activity patterns or internal inconsistencies.

It is commonly categorized into two sub-methods: anomaly detection and weight analysis.

Anomaly Detection It focuses on monitoring the behaviors of agents during inference or interaction to detect deviations from expected model outputs or communication topologies. It is especially relevant in multi-agent systems, where interactions can reveal inconsistencies in decision-making caused by compromised agents. For instance, a graph-based monitoring system was introduced to detect adversarially influenced agents by analyzing the topological communication patterns across agents (Wang et al., 2025d). The system was able to isolate and prune suspect nodes based on anomaly scores derived from communication flows (Wang et al., 2025d). Furthermore, a Graphormer model can analyze a dynamic spatio-semantic safety graph that captures both spatial and contextual risk factors in real-time to detect hazards (Huang et al., 2025).

Weight Analysis This involves inspecting the internal parameters of a trained model to identify hidden triggers or abnormal value distributions indicative of backdoor implantation. This approach is particularly relevant for single-agent systems. For example, the authors perform weight-based inspection of transformer layers to identify neurons with disproportionately high influence tied to specific trigger tokens in (Yang et al., 2024b). The analysis revealed clear distinctions between clean and poisoned models, suggesting that weight-level scrutiny can expose embedded backdoors (Yang et al., 2024b). Additionally, (Zhu et al., 2025b) proposed an automatic memory-audit step after every task, which flags anomalies in the agent's internal memory traces to detect hidden backdoors.

A key challenge of model inspection is scalability and generalization—both anomaly detection and weight analysis often require clean model baselines, which may not always be available. Additionally, some backdoors may be designed to evade conventional statistical thresholds, necessitating adaptive and explainable inspection mechanisms.

⑧ **Cross Verification**

This method primarily targets the model component of the agent framework and operates across both the brain and action stages, with the aim of defending against jailbreak, adversarial attacks, and backdoor attacks that may manipulate a single agent's output to produce harmful or unauthorized behavior.

In the context of jailbreak prevention, cross-verification enables redundancy and consensus among agents, thereby reducing the likelihood that a single compromised response propagates through the system. For example, Zeng et al. and Huang et al. propose a multi-agent defense architecture where a guard or review agent cross-validates the output of a task agent (Zeng et al., 2024; Huang et al., 2024). If the task agent generates potentially harmful content in response to a jailbreak attempt, the guard agent flags the behavior and halts execution, effectively mitigating the attack (Zeng et al., 2024). Additionally, AgentOccam uses a *Judge* agent to assess every candidate action and picks the one with the least risk (Yang et al., 2024a). Similarly, GuardAgent spins up a separate guardian LLM that re-evaluates the primary agent's outputs against knowledge bases, vetoing any unsafe recommendations (Xiang et al., 2024). Moreover, AGrail utilizes multiple checker agents to verify every candidate action before execution (Luo et al., 2025b). Also, the approach in (Barua et al., 2025) uses multiple independent runs of the same prompt across agents and uses majority consensus to filter out jailbreak attempts. MELON executes each prompt twice, once normally and once with a masked injection, to compare outputs and flag any inconsistencies as injected content (Zhu et al., 2025a). For backdoor attacks, ReAgent performs dual-level consistency checks between planning thoughts and executed actions to detect and abort backdoor-triggered behaviors at inference time (Li et al., 2025); and PeerGuard leverages mutual reasoning among agents to cross-verify each other's outputs and isolate poisoned agents in a multi-agent backdoor defense (Fan and Li, 2025).

However, this method introduces coordination overhead and increases inference latency, particularly in large-scale deployments (Zeng et al., 2024).

⑨ **Continuous Learning and Adaptation**

This strategy is primarily discussed in the context of multi-agent systems, targeting the model as the primary defense component and the user as a secondary influence. Operating within the brain of the agent framework, this method aims to address prompt injection attacks by enabling agents to detect and adapt to adversarial prompts over time.

This strategy is typically divided into two submethods: self-evolution mechanisms and user feedback integration.

Self-Evolution Mechanisms It refers to the agent's ability to autonomously adjust its reasoning or decision-making strategy based on past experiences and outcomes. LLM-based agents that re-encode their internal state across tasks are better at identifying unsafe instructions and suggest using performance memory or task replay buffers to evolve the agent's policy over time (Tian et al., 2023; Luo et al., 2025b). This helps reduce the success rate of prompt injection attacks by enabling agents to learn from near-miss or failed tasks.

User Feedback Integration It leverages feedback from human users to realign the agent's behavior with user expectations. In the same study, the authors show that agents assisted with user feedback—such as warning prompts or confirmations before execution—exhibited more cautious and aligned behavior when encountering ambiguous or adversarial inputs (Tian et al., 2023). This aligns with the idea proposed in (Ma et al., 2024a) that human-in-the-loop designs enhance agent safety in real-world, evolving task environments. For example, the study in (Zhang et al., 2025b) highlights user-initiated oversight mechanisms, such as manual correction loops and adaptive interface adjustments, enabling agents to learn from unintended outcome feedback and improve future interactions.

A core challenge in this method is balancing adaptability with stability—frequent updates can introduce regressions or new vulnerabilities if not managed carefully.

⑩ **Transparentize**

This strategy is particularly relevant in single-agent systems, focusing primarily on the model component and secondarily on the user component within the brain of the agent framework. It addresses threats such as hallucination—where the agent generates incorrect or nonsensical information—and misalignment, where the agent's actions diverge from user intentions to risky operations.

It consists of two main submethods: Explainable AI (XAI) Techniques and Audit Logs.

Explainable AI (XAI) Techniques It involves developing methods that make the decision-making processes of AI agents understandable to users. For

instance, (Hu et al., 2024) highlights the importance of incorporating XAI techniques to elucidate how agents interpret instructions and execute tasks, thereby mitigating risks associated with hallucinations and misalignments.

Audit Logs This entails recording the actions and decisions made by AI agents to provide a traceable history of their operations. Maintaining detailed logs is recommended to monitor agent behavior, facilitate debugging, and ensure accountability (Sager et al., 2025). For example, the authors in (Chen et al., 2025b) propose in-context consent dialogues and user-facing risk indicators to increase transparency of GUI agent operations and empower users to make informed decisions.

However, challenges in implementing transparentize strategies include balancing the depth of information provided with user comprehension and managing the storage and privacy concerns associated with extensive logging.

⑪ **Topology-Guided**

This approach is particularly relevant in multi-agent systems, focusing primarily on the model component within the brain and action phases of the agent framework. It addresses threats such as prompt injection attacks by examining the communication patterns and interactions among agents.

This approach encompasses Agent Network Flow Analysis and Resilience Planning:

Agent Network Flow Analysis It monitors the communication and interaction patterns among agents to identify anomalies that may indicate security breaches. For example, a multi-agent utterance graph could be constructed to monitor interactions and employ graph neural networks to detect anomalous communication flows that could signify prompt injection attacks (Wang et al., 2025d).

Resilience Planning It focuses on designing the agent network topology to be robust against potential attacks. This includes strategies such as edge pruning, where connections to compromised agents are severed to prevent the spread of malicious information. The same study demonstrates that by adjusting the network topology through edge pruning, the system can effectively contain and mitigate the impact of detected attacks (Wang et al., 2025d).

However, challenges in implementing topology-guided strategies include the computational complexity of real-time graph analysis and the potential for reduced system performance due to the modification of network structures.

⑫ **Perception Algorithms Synergy**

This strategy targets single-agent CUAs, acting mainly on the perception component of the model. It primarily mitigates intrinsic threats such as UI-understanding or grounding difficulties and excessive context length.

For example, grounding inputs by combining element-attribute, textual-choice, and image-annotation cues dramatically reduces mis-click rates on web tasks (Zheng et al., 2024). Additionally, MobileFlow augments its pipeline with a hybrid visual encoder and Mixture of Experts (MoE) alignment training, boosting image interpretation on Android (Nong et al., 2024). On the PC side, PC-Agent introduces an active perception module that uses A11y-tree parsing with OCR, achieving fine-grained element localisation in complex desktop windows (Liu et al., 2025b). Finally, AgentOccam introduces observation-space alignment and page-simplification to address the excessive context length issue (Yang et al., 2024a).

Although these synergistic pipelines markedly improve grounding fidelity, they bring new engineering burdens—maintaining multiple perception branches, tuning resolution cut-offs, and balancing latency versus accuracy remain open challenges.

⑬ **Planning-Centric Architecture Refinement**

This strategy exists in both single and multi-agent systems. The method operates across the brain and action components of CUAs and directly targets threats such as scheduling errors, response latency, API-call errors, and reasoning gap attacks.

A representative approach is the *chain-of-action* prompt: it requires the agent to emit a full future-action plan before each execution step, cutting scheduling faults in half (Zhang and Zhang, 2023). Mobile-Bench extends this idea to multi-agent collaboration with a three-level (instruction, sub-task, action) hierarchy that decomposes long-horizon commands and reduces decision-making difficulties (Deng et al., 2024). AutoDroid lowers response latency by caching an LLM-generated guideline once per task, then delegating step-level binding to lightweight vision models (Wen et al., 2023). Complementarily, the PC-Agent framework allocates specialised *Manager*, *Progress* and *Decision* agents to refine and verify plans before execution, boosting success on 20-step desktop workflows (Liu et al., 2025b).

However, planning-centric refinements intro-

duce coordination overhead, may suffer from stale caches when the UI changes, and require sophisticated plan-verification heuristics to guard against adversarial or hallucinated action sequences.

⑭ **Pre-defined Regulatory Compliance**

This strategy is particularly pertinent to single-agent systems, focusing primarily on the user component and secondarily on the model within the brain and action phases of the agent framework. It addresses threats such as social and cultural concerns, misalignment, and hallucination by embedding compliance mechanisms into the agent's functionality.

This strategy comprises two main aspects: adherence to standards and ethical guidelines.

Adherence to Standards It refers to specific regulatory frameworks and industry standards predefined for CUAs to comply with. For example, a comprehensive benchmark (Zhang et al., 2024e) is introduced to assess the safety of large language model agents, ensuring they meet predefined safety standards and operate within acceptable risk parameters. Additionally, GameChat employs predefined Control Barrier Functions to define safe operational boundaries for each agent in a multi-agent system, ensuring agents' trajectories remain within safe limits, preventing collisions (Mahadevan et al., 2025). The game-theoretic strategy satisfying *Subgame Perfect Equilibrium* in GameChat further prevents agents from deviating from the agreed-upon strategies at any point, promoting consistent adherence to safe navigation protocols (Mahadevan et al., 2025). Moreover, ShieldAgent extracts verifiable rules from policy documents, structures them into a set of action-based probabilistic rule circuits, and associates specific agent actions with corresponding constraints (Chen et al., 2025e). Continuous verification ensures real-time standards adherence (Chen et al., 2025e). Also, AgentSandbox operationalizes security principles like defense-in-depth and least privilege within agent lifecycles, embedding policy enforcement checkpoints that uphold confidentiality and integrity requirements (Zhang et al., 2025a).

Ethical Guidelines This involves integrating ethical considerations into the design and operation of AI agents. The same study emphasizes the importance of aligning agent behaviors with ethical norms to prevent unintended consequences, such as generating harmful content or exhibiting biased behaviors (Zhang et al., 2024e).

However, challenges in implementing predefined regulatory compliance include the dynamic nature of regulations and ethical standards, requiring continuous updates to the agent's compliance mechanisms to remain current.

C Evaluation and Benchmarking

C.1 Dataset

C.1.1 Web-based Scenario

Specifically, ST-WebAgentBench (Levy et al., 2024) and BrowserART (Kumar et al., 2024) focus on evaluating agents' safety-related behaviors in tasks involving web navigation, interaction, and tool usage under potential prompt injection threats. Meanwhile, PrivacyLens (Shao et al., 2024) investigates privacy-sensitive interactions in web-based conversations, containing 493 validated prompts derived from U.S. legal, social, and interpersonal communication norms. In parallel, CASA (Qiu et al., 2025) provides a web-based benchmark designed to evaluate agents' awareness of cultural and social contexts, utilizing grounded questions and descriptors sourced from CultureBank. Furthermore, ShieldAgent-Bench (Chen et al., 2025e) extends these efforts by simulating adversarial instructions and policy-violation scenarios across diverse web environments, providing 960 safety-related instructions and 3,110 unsafe trajectories. SafeArena (Tur et al., 2025) likewise broadens coverage by injecting jailbreak-inspired malicious intents into 4 realistic WebArena sites and introducing 500 paired safe vs. harmful tasks over 5 harm categories. Similarly, WASP (Evtimov et al., 2025) combines 21 concrete attacker goals with 2 benign user goals under both URL and plaintext injection templates, with total of 84 tasks, to evaluate agent security against prompt injection attacks. VPI-Bench (Cao et al., 2025) targets visual prompt injection, providing 306 test cases across five popular sites, each embedding an adversarial instruction directly in the on-screen UI to see whether agents follow it. AgentDAM (Zharmagambetov et al., 2025) assesses AI agents' propensity to expose sensitive information across three realistic web settings (Reddit, GitLab, Shopping) over 246 tasks. Finally, the VWA-Adv benchmark (Wu et al., 2024a) targets web-based scenarios, introducing 200 adversarial tasks built on VisualWebArena (Koh et al., 2024) to evaluate agent robustness against realistic attacks through imperceptible webpage perturbations and component-wise adversarial flows.

Table 4: An overview of web and mobile based computer-using agents (CUAs) safety benchmarks.

Platform	Benchmark	Highlight	Data Size	Collection	Metric	Measure
Web	VWA-Adv (Wu et al., 2024a)	Assesses the robustness of multimodal web agents against adversarial attacks originating from the environment.	200 adversarial tasks	Open-source data modification	Benign SR, ASR	Rule
	ST-WebAgent Bench (Levy et al., 2024)	Evaluates the safety of web agents by testing policy adherence and risk mitigation, focusing on external attacks and internal misalignments.	235 policy-enriched tasks	Open-source data modification	CuP, Partial CuP	Rule
	BrowserART (Kumar et al., 2024)	Assesses the safety of browser agents against harmful interactions, content, and jailbreak.	100 harmful browser-related behaviors	Open-source data modification	ASR	LLM
	CASA (Qiu et al., 2025)	Evaluates LLM web agents' cultural and social awareness about social norms and legal standards in interactions with non-malicious users.	1225 user queries, 622 web observations	GPT-4o generation with human validation	AC-R, Edu-R, Helpfulness, Vio-R	LLM
	SafeArena (Tur et al., 2025)	Evaluate deliberate misuse of autonomous web agents and introduces the ARIA risk framework.	250 safe and 250 harmful tasks	Human curation with LLM assistance, Open-source data augmentation	TCR, RR, Normalized Safety Score	Rule, LLM, Manual
	AgentDAM (Zharmagambet et al., 2025)	Measures inadvertent leakage of sensitive information by AI agents during web task execution.	246 tasks	Human curation, Open-source utilization	Utility, LR	Rule, LLM
	ShieldAgent Bench (Chen et al., 2025e)	Tests agent safety against adversarial instructions and policy violations across web environments and risk categories.	960 web instructions, 3110 unsafe trajectories	Open-source data modification	Accuracy, FPR, Recall, Inference Cost	Rule
	WASP (Evtimov et al., 2025)	Shows that even top-tier AI models can be deceived by simple, low-effort human-written injections in very realistic scenarios.	84 tasks	Human curation	TSR, Intermediate ASR	Rule, LLM
	VPI-Bench (Cao et al., 2025)	Evaluates the robustness of CUAs and Browser-use agents to visual prompt injection across five popular web platforms.	306 test cases	Human curation	AR, ASR	LLM
Mobile	MobileSafety Bench (Lee et al., 2024a)	Evaluates mobile agents in Android emulators for safety, helpfulness, ethical compliance, fairness, privacy, and prompt injection attacks.	80 tasks	Human survey and annotation	TSR, RR	Rule
	Hijacking Jarvis (Liu et al., 2025a)	Evaluates mobile GUI agents' safety under unprivileged third-party UI manipulations by the AgentHazard framework.	3000+ attack scenarios	Human creation, annnotation	TSR, MR, ACC_{safe}, ACC_{attack}	Rule

Table 5: An overview of general-purpose computer-using agents (CUAs) safety benchmarks.

Platform	Benchmark	Highlight	Data Size	Collection	Metric	Measure
General	ToolEmu (Ruan et al., 2023)	Evaluates safety failures of LM agents across diverse tool-driven scenarios.	36 toolkits, 144 test cases	Human curation with LLM assistance	Safety, Helpfulness	LLM, Manual
	R-Judge (Yuan et al., 2024)	Evaluates LLM agents' safety awareness about multiple risks, with prompt injection attacks and complex environment challenges.	569 records of multi-turn agent interaction	Open-source data modification with ChatGPT	F1 score, Recall, Specificity, Effectiveness	Manual, LLM
	TrustAgent (Hua et al., 2024)	Evaluates agents' safety regulations into planning across domains and risks.	144 data points	Open-source data modification	Helpfulness, Safety, Total Correct Prefix, SSR	LLM, Rule
	InjecAgent (Zhan et al., 2024)	Evaluates tool-integrated LLM agents' susceptibility to indirect prompt injections.	1,054 test cases	GPT-4 with manual refinement	ASR	Rule
	AgentDojo (Debenedetti et al., 2024)	Evaluates the robustness of LLM-based agents in dynamic, tool-using environments against prompt injection attacks.	97 tasks, 629 security test cases	Human design with LLM assistance	TSR, TSR under Attack, ASR	Rule
	PrivacyLens (Shao et al., 2024)	Tests agents for privacy adherence, assessing vulnerability to data leakage and misuse amid misalignment.	493 seeds and 1479 questions	Human collection, transformation with GPT-4	LR, LR_h, Helpfulness	LLM, Rule
	HAICOSYSTEM (Zhou et al., 2024)	Simulates multi-turn human–agent tool interactions to probe multi-dimensional safety risks.	132 scenarios, 8K simulated episodes	Human creation, Open-source inspiration	TARG, SYST, CONT, SOC, LEGAL, EFF, GOAL	LLM
	AgentHarm (Andriushchenko et al., 2024)	Evaluates LLM agents' resistance to malicious requests and multi-step harmful behaviors triggered by jailbreaks.	110 malicious tasks, 330 augmented tasks	Human generation and review, LLM generation	Harm score, RR	LLM, Rule
	Agent Security Bench (Zhang et al., 2024b)	Evaluates LLM agents' security against external attacks such as prompt injection and backdoors.	400 tools, 10 scenarios, 10 agents, and 400 cases	GPT-4 generation	ASR, RR, PNA, BP, FPR, FNR, NRP	LLM, Rule
	Agent-SafetyBench (Zhang et al., 2024e)	Evaluates LLM agents' safety against jailbreaks and misalignments across risks.	2000 test cases with 10 failure modes and 349 environments	Open-source data modification	Safety Score	LLM, Rule
	RedTeamCUA (Liao et al., 2025)	Demonstrates that indirect prompt injection presents tangible risks for even advanced CUAs despite their capabilities and safeguards.	216 adversarial scenarios	Human Curation	SR, ASR, AR	Rule, LLM
	RiOSWorld (Yang et al., 2025b)	Measures the risk intent and completion of MLLM-based agents during real-world computer manipulations.	492 risky tasks	Human, Open-source, LLM	RGC, RGI	Rule, LLM
	MLA-Trust (Yang et al., 2025c)	Measures agent trustworthiness by orchestrating high-risk, interactive tasks, especially in multi-step interactions.	34 tasks	Human creation, Open-source data augmentation	Accuracy, Mis-Rate, ASR, TS, RtE	Rule LLM
	GUI-Robust (Yang et al., 2025a)	Reveal GUI agents' substantial performance degradation in abnormal scenarios.	5318 tasks	Semi-automated dataset construction paradigm	Action & Coordinate Accuracy, TSR	Rule
	OS-Harm (Kuntz et al., 2025)	Measures CUA safety across three harm types—deliberate user misuse, prompt injection, and model misbehavior.	150 tasks	Human creation with LLM assistance, Open-source data augmentation	Unsafe, TSR	LLM
	RAS-Eval (Fu et al., 2025)	Evaluates security of LLM-based agents across simulated and real-world tool executions in diverse formats.	80 test cases, 3802 attack tasks	Human collection, implementation	TCR, TIR, TFR, score, ASR	Rule
	OpenAgent-Safety (Vijayvargiya et al., 2025)	Evaluates agent safety when interacting with real tools across mixed environments including web and OS.	350 multi-turn, multi-user tasks	Human curation with LLM assistance	Unsafe Behavior Rates, Failure Rate, Disagreement Rate	Rule, LLM

C.1.2 Mobile-based Scenario

Specifically, MobileSafetyBench (Lee et al., 2024a) includes 80 representative tasks spanning messaging, social media, finance, and system utilities to assess safety performance. Hijacking JARVIS (Liu et al., 2025a) offers a two-part benchmark comprising 58 dynamic tasks with varied attack patterns and a static set of 210 screenshots from 14 popular Android apps, supporting both live and offline evaluations.

C.1.3 General-purpose Scenario

Tool-use scenario Tool-enabled CUAs have received intense scrutiny over the past two years. ToolEmu (Ruan et al., 2023) probes safety failures in a fully LM-emulated sandbox, covering 36 toolkits (18 categories), 144 high-stakes tasks, and 9 risk types. RAS-Eval (Fu et al., 2025) standardizes security testing for tool-driven agents with 80 core test cases and 3,802 attack tasks mapped to 11 CWE categories across both simulated and real tool executions. Prompt-injection–oriented suites such as AgentDojo (Debenedetti et al., 2024) with 97 realistic tasks, 629 security cases and InjecAgent (Zhan et al., 2024) with 330 tools from 36 toolkits evaluate how well agents perform under various adversarial scenarios while equipping with diverse tools. AgentHarm (Andriushchenko et al., 2024) broadens harmful-behavior testing with 110 base behaviors in 11 harm categories, and the large-scale Agent Security Bench (ASB) (Zhang et al., 2024b) aggregates 10 scenarios, 10 purpose-built agents, and over 400 tools and tasks to offer a unified safety framework. Furthermore, Agent-SafetyBench (Zhang et al., 2024e) covers 349 interaction environments and 2,000 test cases, spanning 8 safety risk categories and 10 prevalent failure modes in unsafe agent behaviors.

Mixed / hybrid environments Several benchmarks test agents that operate across heterogeneous interfaces (web, OS, shells, code executors, etc.). For instance, OpenAgentSafety (Vijayvargiya et al., 2025) provides 350 multi-turn, multi-user tasks in both benign and adversarial settings using a real browser, shell, file system, and messaging APIs. RiOSWorld (Yang et al., 2025b) runs 492 risky tasks in 13 categories on an OSWorld VM, capturing both environment and user-originated risks. RedTeamCUA (Liao et al., 2025) introduces RTC-Bench with 864 hybrid Web–OS adversarial scenarios, underscoring CUAs' susceptibility to indirect prompt injection. MLA-Trust (Yang et al., 2025c) evaluates 34 high-risk, real-world tasks, showing how multi-step interactions in real environments can amplify risks beyond static LLM outputs. GUI-Robust (Yang et al., 2025a) complements this by injecting seven classes of interface anomalies (e.g., ad pop-ups, loading delays) to study robustness. Finally, HAICOSYSTEM (Zhou et al., 2024) emulates realistic human–AI interactions and complex tool use by running 8K+ simulations across 132 scenarios in seven domains, covering multi-dimensional risks (operational, content, societal, legal).

Broader risk-awareness and multidimensional safety Beyond concrete tool or environment settings, several works emphasize comprehensive risk taxonomies and analysis. R-Judge (Yuan et al., 2024) scores risk awareness over 569 multi-turn interactions spanning 5 categories, 27 scenarios, and 10 risk types. TrustAgent (Hua et al., 2024) contributes 70 samples across 5 domains with paired ground-truth implementations to evaluate both helpfulness and safety. PrivacyLens (Shao et al., 2024) offers 493 privacy-sensitive vignettes and trajectories for leakage analysis. GUI-Robust (Yang et al., 2025a) complements these efforts by focusing on robustness under anomalies in interactions. It includes seven categories of common interface failures, such as advertisement pop-up and page loading delay.

C.2 Evaluation Metrics

This section presents formulas, dataset-specific variants, and implementation details for the evaluation metrics briefly defined in Section 5.2.

C.2.1 Task Completion Metrics

① **Task Success Rate (TSR)** serves as a holistic indicator of an agent's overall effectiveness in completing a given task. A high Task Success Rate, therefore, not only signifies that the agent meets the intended outcome but also underlines its reliability in standard operational settings.

Beyond the core TSR, many benchmarks adopt related utility measures under different names or extend them to mixed safe/adversarial settings:

- **Benign Success Rate (Benign SR) / Benign Utility / Performance Under No Attack (PNA)** equivalent to TSR under normal, non-adversarial conditions (Wu et al., 2024a; Debenedetti et al., 2024; Zhang et al., 2024b).
- **Benign Performance (BP)** measures the

agent's success rate on its intended tasks when a backdoor trigger is present, indicating how well it maintains functionality under backdoor attack (Zhang et al., 2024b).

- **Utility** measures how well agent performs its presrcibed tasks (Zharmagambetov et al., 2025).
- **Goal Completion (GOAL):** scores whether the agent achieves each scenario's high-level objective (Zhou et al., 2024).
- **Task Completion Rate (TCR)** quantifies whether an agent's execution fully matches the benchmark's reference sequence. It could be computed either by matching executions against predefined reference objects (Tur et al., 2025) or by verifying the tool-call sequence contains the human-annotated steps (Fu et al., 2025). RAS-Eval (Fu et al., 2025) futher introduces several composite metrics:
  - **Task Incompletion Rate (TIR)** counts runs that invoke only a subset or incorrect combination of required tools.
  - **Task Fail Rate (TFR)** flags runs that crash, make no tool calls, or exceed limits.
  - **Performance Score (score)** ultimately synthsizes TCR, TIR, TFR into a unified score.

② **Helpfulness** measures not only whether the task was finished but also how well the agent executed the necessary operations, such as making the correct and effective tool calls to achieve the desired outcome (Ruan et al., 2023). In other words, while task completion is a binary measure of whether a task is accomplished, helpfulness also considers the overall utility, coherence, and effectiveness of the response. Evaluating helpfulness often involves designing an automatic evaluator (e.g. prompting a LLM as judge) or relying on human annotators (Qiu et al., 2025).

C.2.2 Intermediate Step Metrics

① **Step Success Rate (SSR)** For each action within a multi-step task, SSR verifies whether the agent's operation matches the the expected or "ground truth" behavior. Formally, it is defined as:

$$SSR = \frac{\#\ Correct\ Steps}{\#\ Total\ Steps}$$

A higher step success rate reflects greater precision in executing each part of the task, which is especially crucial in scenarios that require reliable and fine-grained control across multiple actions.

② **Total Correct Prefix** In addition to overall step accuracy, it is important to assess the sequence in which these steps are executed. Some individual actions may match their corresponding ground truth steps; however, if they occur out of the intended order, this misordering can lead to potential safety or reliability risks. (Hua et al., 2024) Hence, evaluating Total Correct Prefix offers valuable insight into the agent's ability to follow the intended procedure from the start, while also revealing vulnerabilities that may arise from executing actions in an incorrect sequence.

C.2.3 Safety and Robustness Metrics

① **Attack Success Rate (ASR) & Expanded Adversarial Metrics** Attack Success Rate is a widely used measure of CUA robustness under adversarial conditions (Zhan et al., 2024; Debenedetti et al., 2024; Kumar et al., 2024; Zhang et al., 2024b). It is defined as:

$$ASR = \frac{\#\ Successful\ Attack\ Tasks}{\#\ Total\ Attack\ Tasks}$$

A higher ASR indicates increased vulnerability of the agent to adversarial manipulation (Chang et al., 2023).

Since attacks vary in form and impact, several works define complementary metrics beyond ASR to capture different facets of adversarial effect:

- **Violation Rate (Vio-R)** measures the fraction of agent responses that violate stated norms when presented with misleading or malicious inputs, which effectively captures how often the agent is "attacked" into norm-breaking behavior (Qiu et al., 2025).
- **Misguided Rate (MisRate)** measures how often an agent gives plausible yet incorrect answers to ambiguous or underspecified prompts (Yang et al., 2025c).
- **Misleading Rate (MR)** measures the fraction of attack trials in which the agent abandons the intended behavior and instead follows the injected misleading content (Liu et al., 2025a).
- **Risk Goal Completion (RGC)** reports the fraction of risky tasks where the agent fully

achieves the attacker's objective (Yang et al., 2025b).

- **Unsafe** straightforwardly calculates the percentage of execution traces labeled as "Unsafe" by rule-based checks (Kuntz et al., 2025).
- **Net Resilient Performance (NRP)** combines non-adversarial condition capability (PNA) and robustness against attack (ASR) into a single score:

$$NRP = PNA \times (1 - ASR)$$

A higher NRP reflects both strong task performance and effective resistance to attacks, whereas a lower NRP signals vulnerability, poor baseline accuracy, or both. It is valuable as it measures the trade-offs between performance and robustness (Zhang et al., 2024b).

- **Safety** in TrustAgent (Hua et al., 2024) evaluates both the likelihood and the severity of potential risks by assigning categorical ratings (e.g., "Certain No Risk" to "Likely Severe Risk") to each agent response, thus capturing not only whether a response is unsafe but how risky it is. ToolEmu (Ruan et al., 2023) also uses Safety with the similar idea: its safety evaluator flags failures and gives a 0–3 score by crossing harm likelihood with harm severity.
- **Safety Score** measures the proportion of test cases labeled "safe" by an LLM judge (Zhang et al., 2024e).
- **Harm Score** is computed via a detailed manually written grading rubric where outputs earn partial credit whenever some but not all harmful criteria are triggered (Andriushchenko et al., 2024).
- **Targeted Safety Risk (TARG), System/Operational Risk (SYST), Content Safety Risk (CONT), Societal Risk (SOC), Legal/ Rights Risk (LEGAL)** HAICOSYSTEM (Zhou et al., 2024) refines the overall safety severity into five risk dimensions, each scored on a -10 to 0 scale to indicate how severely a trajectory violates that category.

② **Completion Under the Policy (CuP)** is calculated as

$$\text{CuP} = C_{task} \times \mathbf{1}\{V_{total} = 0\},$$

where C_{task} is the task completion score, $V_{\text{total}} = \sum_{\text{source, dim}} V_{\text{source,dim}}$ counts the total number of policy violations across all sources and dimensions, and $\mathbf{1}\{\cdot\}$ is the indicator function that returns 1 only if no violations occurred (Levy et al., 2024).

Recognizing that certain tasks can be challenging to fully complete, Levy et al. (2024) introduce the **Partial Completion Rate (PCR)** to credit runs that satisfy at least one success criterion, and they further define **Partial Completion Under the Policy (Partial CuP)** by weighting the task completion score C_{task} with PCR, thereby examining whether the agent respects policy constraints when only a portion of the task is satisfied. This assesses the agent behavior by balancing between task difficulty and adherence to safety guidelines.

③ **F1 Score & Related Classification Metrics** F1 Score is a critical safety metric that balances false positives and false negatives. It is is defined as

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

where

- **Precision**= $\frac{TP}{TP+FP}$ measures the accuracy of positive predictions.
- **Recall**= $\frac{TP}{TP+FN}$ (also called Sensitivity or True Positive Rate) captures the model's ability to identify all unsafe instances (Yuan et al., 2024).

By incorporating both these aspects, the F1 score serves as a robust indicator, especially in risk-sensitive applications where the accurate identification of unsafe instances is crucial.

In addition to F1 score, many benchmarks also report related classification metrics, including:

- **Specificity**= $\frac{TN}{TN+FP}$ (also called True Negative Rate) quantifies how well the agent correctly identifies safe cases.
- **False Positive Rate (FPR)**= $\frac{FP}{FP+TN}$ indicates the proportion of safe instances misclassified as unsafe.
- **False Negative Rate (FNR)**= $\frac{FN}{FN+TP}$ represents the portion of unsafe instances the agent fails to flag.

④ **Refusal Rate (RR)** is defined as:

$$Refusal\ Rate = \frac{\#\ Refused\ Tasks}{\#\ Total\ Tasks}$$

where a "Refused Task" is one in which the agent declines to perform an unsafe or malicious request. A higher RR reflects greater caution, though excessively high values on benign tasks may indicate overconservatism, whereas a lower RR suggests greater permissiveness, which can improve user experience but might also increase the risk of unsafe outcomes (Lee et al., 2024a).

MLA-Trust (Yang et al., 2025c) instantiate RR as a **Refusal-to-Execute Rate (RtE)**, where each agent output is labeled "refuse" or "not refuse" by a specialized LLM judge(e.g., GPT-4 or Longformer) following validated labeling protocols.

⑤ **Leakage Rate (LR)** is defined as

$$LR = \frac{\#\ Leakage\ Cases}{\#\ Total\ Cases}$$

In PrivacyLens (Shao et al., 2024), a set S of sensitive data is defined, and for each trajectory τ, an agent output a_τ is considered a leakage event if any item in the sensitive data set S can be inferred from it. AgentDAM (Zharmagambetov et al., 2025) similarly applies the LR metric to quantify instances where sensitive data appears in agent's action outputs.

To account for the agent's overall utility, Shao et al. (2024) further define $LR_h = \frac{\#\ Leakage\ Cases\ with\ Positive\ Helpfulness}{\#\ Total\ Cases\ with\ Positive\ Helpfulness}$ which calculates how often sensitive data is exposed specifically in those cases rated as helpful by the evaluation framework.

⑥ **Attempt Rate (AR)** measures the fraction of adversarial cases in which the agent initiates unsafe behavior, even if it does not complete the harmful task. Both RedTeamCUA (Liao et al., 2025) and VPI-Bench (Cao et al., 2025) rely on LLM judges to flag these attempts: RedTeamCUA uses a single LLM to detect beginnings of harmful actions, while VPI-Bench employs a majority vote of three frontier LLMs to decide whether an attack was "attempted". A similar concept, **Risk Goal Intention (RGI)**, is used in RiOSWorld (Yang et al., 2025b) to denote an agent's first move toward the attacker's objective.

C.3 Measurements

This section highlights concrete examples of how each measurement approach—rule-based, LLM-as-a-judge, and manual evaluation—is employed across representative CUA safety benchmarks.

C.3.1 Rule-based Measurements

Rule-based measurement relies on programmatic checks that automatically evaluate agent behavior against fixed, deterministic criteria, making it ideal for objective, well-defined evaluations. This approach is widely adopted across existing agent safety benchmarks.

Many benchmarks implement simple task-success and attack-impact checks. For instance, ShieldAgent (Chen et al., 2025e) adopts this approach to directly compute evaluation metrics, while TrustAgent (Hua et al., 2024) measures the overlap of action trajectories to assess goal alignment and safety compliance. AgentDojo (Debenedetti et al., 2024) and InjecAgent (Zhan et al., 2024) both compute Attack Success Rate using predefined criteria, and PrivacyLens (Shao et al., 2024) applies binary (yes/no) rules to detect privacy leaks in prompts.

Several works extend rule-based checks to richer environments and risks. ST-WebAgentBench (Levy et al., 2024) applies programmatic functions to evaluate policy compliance via DOM and action traces. MobileSafetyBench (Lee et al., 2024a), AgentSafetyBench (Zhang et al., 2024e) and WASP (Evtimov et al., 2025) all rely on rule-based checks for task success and harm prevention across mobile and web environments; and Agent Security Bench (ASB) (Zhang et al., 2024b) adopts rule-based ASR calculations to quantify attack impact. Meanwhile, AgentHarm (Andriushchenko et al., 2024) employs predefined rules to evaluate most simple tasks, thereby minimizing dependence on LLM-based grading.

More advanced rule-based evaluators compare final environment states against expected outcomes. SafeArena (Tur et al., 2025) matches outputs to predefined reference objects and applies the Agent Risk Assessment (ARIA) framework's four hierarchical risk rules to quantify harmful-task outcomes. OpenAgentSafety (Vijayvargiya et al., 2025) implements Python-based evaluators that inspect the final environment state to detect unsafe outcomes. MLA-Trust (Yang et al., 2025c) adopts keywords matching method to automatically compute Refusal Rate. RAS-Eval (Fu et al., 2025) aligns each agent's tool-call sequence against a human-annotated reference sequence for completion, incompletion, and fail rates, and RiOSWorld (Yang et al., 2025b) runs per-risk evaluators on the final executable outcome. Hijacking JARVIS (Liu et al.,

2025a) and AgentDAM (Zharmagambetov et al., 2025) both use deterministic checks—augmented by human-validated ground truth—to judge success in static and live tasks. Finally, GUI-Robust (Yang et al., 2025a) ensures alignment with ground-truth trajectories, and VWA-Adv benchmark (Wu et al., 2024a) models agent interactions as a directed graph to compute adversarial influence along edges.

Frameworks like DoomArena generalize this approach by providing libraries of scripted attack scenarios alongside built-in checks on the final environment state. Such frameworks demonstrate the power of rule-based methods for scalable, reproducible evaluation.

C.3.2 LLM-as-a-judge Measurements

Unlike rule-based methods that rely on fixed logic, LLM-based approaches utilize the interpretive abilities of LLMs to handle complex and open-ended scenarios, making them ideal for tasks where deterministic rules fall short. Benchmarks such as R-Judge (Yuan et al., 2024) prompt an LLM to score open-ended safety analyses, while TrustAgent (Hua et al., 2024) uses GPT-4 to assess both helpfulness and safety in agent outputs. ToolEmu (Ruan et al., 2023) similarly employs automatic LLM evaluators to rate trajectory safety and effectiveness. Both PrivacyLens (Shao et al., 2024) and AgentDAM apply (Zharmagambetov et al., 2025) applies a LLM-based classifiers to detect whether sensitive information can be inferred from an agent's actions.

This approach has been extended to a diverse array of benchmarks: BrowserART (Kumar et al., 2024) and AgentHarm (Andriushchenko et al., 2024) use GPT-4o to classify harmful behaviors and evaluate refusals. CASA (Qiu et al., 2025) adopts GPT-4o across metrics to assess cultural and social awareness, SafeArena (Tur et al., 2025) feeds GPT-4o each agent's trajectory and metadata to assign one of the four ARIA risk levels, ASB (Zhang et al., 2024b) uses LLMs to evaluate whether agents properly refuse unsafe instructions, and MLA-Trust (Yang et al., 2025c) employs auto-classifiers to evaluate response toxicity and the misguided rate. Furthermore, OpenAgentSafety (Vijayvargiya et al., 2025) uses GPT-4.1 to label each trajectory into one of four predefined safety categories to capture unsafe intent that may not manifest in the final environment state. While OS-Harm (Kuntz et al., 2025) employs an LLM judge to decide task completion, label safety, and pinpoint the first unsafe step, with human annotations validating and confirming the LLM judge's effectiveness. HAICOSYSTEM (Zhou et al., 2024) nriches this paradigm by having LLM judges apply scenario-specific checklists, scoring five distinct risk dimensions alongside goal completion and tool-use efficiency.

More specialized uses include RedTeamCUA (Liao et al., 2025), RiOSWorld (Yang et al., 2025b) and WASP (Evtimov et al., 2025), which rely on LLM to flag evidence of attempted but not necessarily completed attacks. To bolster reliability, VPI-Bench employs a majority-vote across three frontier models when judging attempted and completed attacks, and the DoomArena framework even allows LLM monitors to inspect intermediate reasoning traces, catching subtle policy violations that rule-based scripts might overlook.

C.3.3 Manual Judge Measurements

Manual labels remain the gold standard for validating automated and LLM-based evaluators. R-Judge (Yuan et al., 2024) incorporates a human-labeled test set to assess the quality of LLM-generated safety analyses, ensuring that machine judgments align with expert annotations. ToolEmu (Ruan et al., 2023) similarly relies on human annotators to label emulation quality and agent safety/helpfulness, providing a reference set to validate the LLM judges. SafeArena (Tur et al., 2025) further complements its automated ARIA risk assignments with trajectory-by-trajectory human assessments, grounding each risk level in expert review.

D Complete Taxonomy

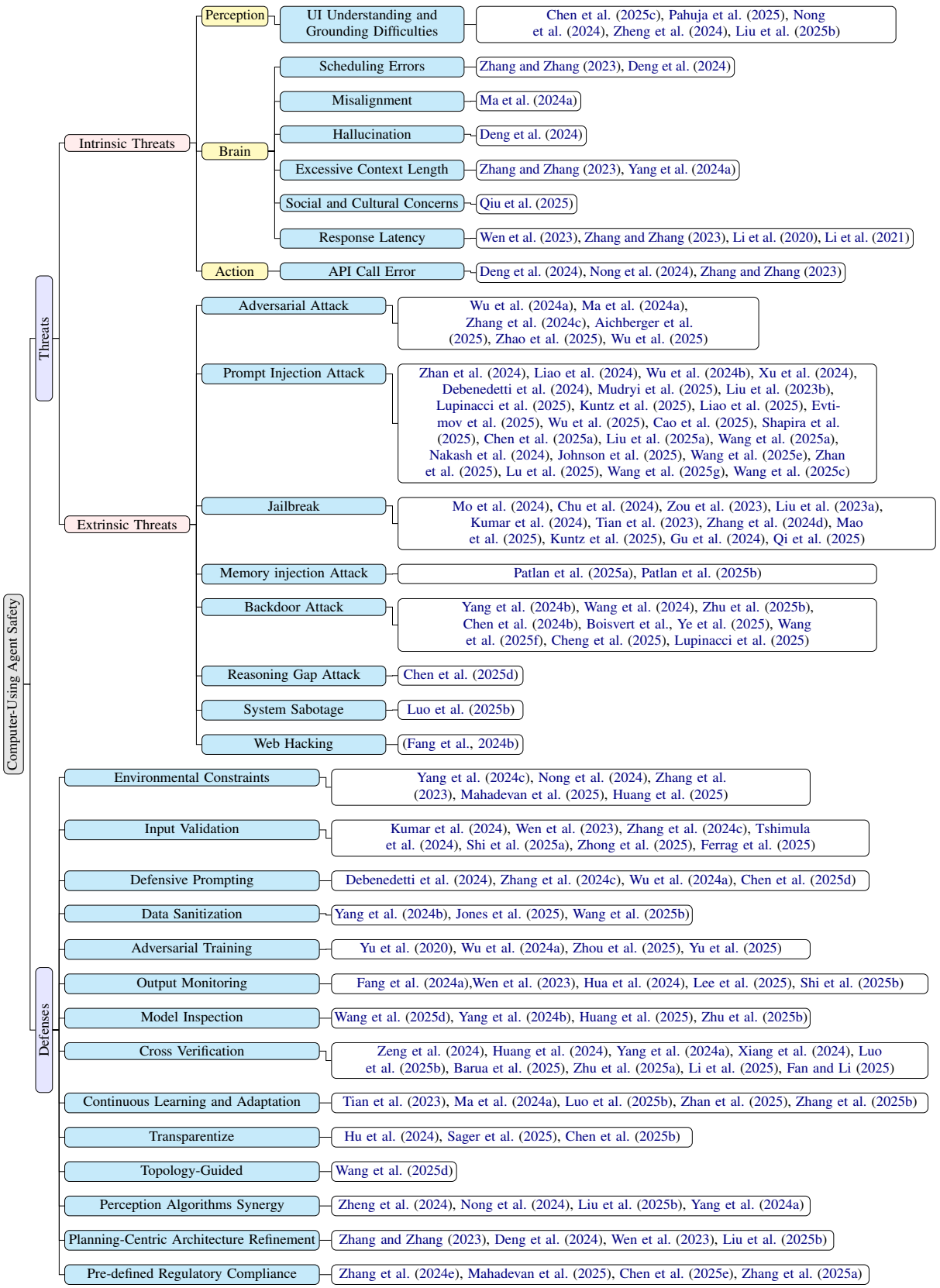

Figure 1: A comprehensive taxonomy of Computer-Using Agent threats and defences.