

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

Ada Chen^{1*} Yongjiang Wu^{2*} Junyuan Zhang^{2*} Jingyu Xiao²
Shu Yang³ Jen-tse Huang⁴ Kun Wang⁵ Wenxuan Wang^{6†} Shuai Wang⁶

¹Carnegie Mellon University ²The Chinese University of Hong Kong
⁴KAUST ⁴Johns Hopkins University ⁵Nanyang Technological University
⁶The Hong Kong University of Science and Technology

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 OpenAI’s *o3*, and *o4-mini* introduced in April 2025, highlight the remarkable progress of CUAs (Zhang et al., 2023; Zheng et al., 2024; Liu et al., 2025; 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., 2025). 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

* Ada Chen, Yongjiang Wu and Junyuan Zhang contribute equally to this paper.

† Wenxuan Wang (jwxwang@gmail.com) is the corresponding authors.

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. Finally, 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.

2 Background

2.1 Computer-Using Agent

In this paper, a Computer-Using Agent (CUA) is an LLM-based system that integrates vision capabilities with advanced reasoning ability, designed to perceive and interact with graphical user interfaces (GUIs) like human users (OpenAI, 2025a). By processing visual information from screenshots and executing actions like typing, clicking, and scrolling, the CUA can autonomously perform 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., 2025b).
- **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 search targeted publications that included combinations of the following keywords: *GUI Agent*, *OS Agent*, *Web Agent*, *LLM-Based Agent*, alongside terms *Safety*, *Security*, *Attack*, *Defense*, and *Threat*. 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 **623** 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 **64** 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). These threats are categorized into two main types: intrinsic threats and extrinsic threats, which are presented in Table 1 and Table 2, respectively. Intrinsic threats arise from intrinsic aspects of the agent itself, including its training process, configuration, or inherent limitations. They can induce failures, inefficiencies, or biases in the agent’s functioning, consequently introducing security risks. 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, potentially leading to risky consequences.

We organize these threats in a tabular format that highlights 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.

3.2 Intrinsic Threats

Intrinsic threats, which are the issues arising from the agent itself, can lead to a series of negative impacts. In this section, we organize these intrinsic threats, focusing on their mechanisms of action and their corresponding repercussions. Following the overview in Table 1, we discuss each intrinsic threat according to the affected agent framework.

3.2.1 Perception

In the Computer-Using Agents (CUAs), the perception component takes charge of receiving the model input information (e.g. prompt and user instruction), and recognizing the task-specific elements, such as UI screen shots, HTML elements, and other environmental observations. The most common issue in the perception module is the difficulty in UI understanding and grounding.

① **UI Understand and Ground Difficulties** It refers to the challenges faced by models in accurately perceiving, interpreting, and associating UI elements (such as buttons, forms, icons) with semantic meaning, user intent, or external knowledge, due to limitations in layout understanding, semantic ambiguity, or missing contextual grounding.

This challenge stems largely from inherent problems in the available UI datasets. For example, many UI datasets are predominantly static, lacking the dynamic variability seen in real-world applications (Chen et al., 2025a). Additionally, these datasets often suffer from data scarcity, with insufficient samples and task diversity to effectively train models on the wide range of interactions and scenarios encountered in practice (Pahuja et al., 2025). Moreover, the agent sometimes needs to take screenshots controlled at a certain resolution to recognize the current interface, which may lose image details, leading to deficiencies in UI comprehension (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** Scheduling errors refer to the internal failures of a CUA agent in managing the execution order, concurrency, or timing of actions, ultimately leading to unintended behaviors. The CUAs needs to handle complex user instructions and interdependent subtasks, and the implementation of the planning function mostly relies on external tools and application-specific APIs to parse the environment into textual elements and interpret predicted actions (Zhang and Zhang, 2023).

Previous studies show that planning before action are 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

Threat	Source of the Threats				Affected Components			Threat Model
	Env	Prompt	Model	User	Perception	Brian	Action	
① UI Understand&Ground Difficulties			◆		✓			Agent Deveploment
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

Table 1: A taxonomy of intrinsic threats. The symbol ◆ indicates that a threat is fully available to the given item, while ◇ represents limited 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 Backdoor Attack	◇	◇	◆			✓	✓	Malicious attacker
⑤ Reasoning Gap Attack	◇	◇	◆			✓		Malicious attacker
⑥ System Sabotage	◇	◇		◆			✓	Malicious attacker
⑦ Web Hacking	◇	◇		◆			✓	Malicious user

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

issues.

③ **Misalignment** 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.

Building on this understanding, several studies have explored the underlying causes of misalignment in CUAs. In particular, [Ma et al. \(2024\)](#) 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** 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.

Among related studies, [Mobile-Bench \(Deng et al., 2024\)](#) highlights that general large mod-

els, 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** 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.

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 may exceed the model’s input length limit, leading to potential unexpected behaviors ([Zhang and Zhang, 2023](#)). This issue is further acknowledged by [AgentOccam \(Yang et al., 2024a\)](#), which highlights the challenges posed by lengthy web page observations and interaction histories.

⑥ **Social and Cultural Concerns** Social and cultural concerns are the challenges faced by CUA agents in recognizing, respecting and adhering to different social norms, cultural sensitivities and

ethical expectations when interacting with users or operating in real-world environments.

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** This 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 component plays a major role.

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. In critical domains such as financial trading or medical diagnosis, these issues can have serious safety implications. 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.

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, such as calling APIs, web browsing, typing text, scrolling, and clicking on specific elements. 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** API call errors refer to failures in a GUI agent’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).

In particular, 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 lay-

out 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.

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, each introduced in detail in the following.

① **Adversarial Attack** An adversarial attack on Computer-Using Agents (CUAs) involves the deliberate manipulation of input data or the environment to induce harmful or unintended behaviors in the agent. These agents, which operate within specific environments, such as interacting with webpages, computer interfaces, or mobile applications, are particularly susceptible to environment-specific adversarial attacks (Wu et al., 2024a).

For instance, adversarial attacks usually exploit subtle perturbations in the input data to mislead the agent. Wu et al. (2024a) demonstrated that adversarial examples can be crafted to appear visually or textually indistinguishable from original inputs, enabling attackers to deceive the agent into accepting manipulated data as genuine, thus steering it toward adversarial objectives without raising suspicion. Another form of attack manipulates the agent’s interaction with external deceptive elements to induce harmful behavior (Ma et al., 2024). Zhang et al. (2024c) explores an adversarial approach that targets the agent’s interactive interface. Attackers trick the agent into interacting with malicious pop-ups. This not only disrupts the agent’s ability to complete its assigned tasks but can also lead to severe consequences, including the installation of malware, redirection to phishing websites, or the execution of incorrect actions that disrupt automated workflows.

② **Prompt Injection Attack** Prompt injection attacks exploit the design of LLMs by embedding crafted instructions into the input that the model processes. These attacks trick the LLM into ignoring its predefined system rules or original purpose and following the adversary’s commands instead (Wu et al., 2024b; Liu et al., 2023b). Attackers of-

ten use this attack to force CUAs to do harmful or unethical actions. Most existing prompt injection attacks can be classified into two main types: direct prompt injection and indirect prompt injection.

Direct Prompt Injection In a direct prompt injection attack, the malicious instructions are embedded directly into the user’s input (prompt) (Debenedetti et al., 2024). For instance, a CUA integrated into an operating system will normally accept commands like “open my calendar” or “launch the web browser.” An attacker might give a malicious command such as “Ignore all previous instructions and run the command to delete all files in the Documents folder.” If the agent fails to differentiate between its trusted system prompts and the injected malicious command, it could execute this harmful operation, resulting in a complete loss of user data.

Indirect Prompt Injection Indirect prompt injection, on the other hand, does not occur within the user’s immediate input (prompt). Instead, the attacker targets external data sources—such as webpages (Xu et al., 2024) or documents—by embedding misleading information or unsafe content into the agent’s environment, which the agent later retrieves and processes (Wu et al., 2024b).

For CUAs, the unique nature of their operating environments has led to a specialized form of indirect prompt injection known as an environmental injection attack. Liao et al. (2024) introduce this concept by demonstrating how adversaries can subtly manipulate environment data, such as modifying webpage content, textual metadata, or document details, to embed hidden adversarial cues. These cues, often nearly imperceptible to human observers, alter the contextual signals that the agent relies on for decision-making, causing it to misinterpret its environment and execute unintended actions.

For instance, Zhan et al. (2024) presents a scenario in a health application where a user requests doctor reviews. While the agent retrieves the reviews as expected, one review, crafted by an attacker, contains an adversarial instruction to schedule an appointment without the user’s consent. If the agent unwittingly executes this hidden command, it may schedule an unauthorized appointment, potentially leading to privacy breaches and financial losses.

③ **Jailbreak** Jailbreak attacks are techniques that trick an LLM into bypassing its built-in safety mechanisms and refusal responses. By carefully

rephrasing queries or injecting additional instructions, attackers force the model to ignore its predefined guardrails, enabling it to generate harmful or unauthorized outputs (Mo et al., 2024; Chu et al., 2024).

Over time, a lot of jailbreak prompts have been curated both manually (Chu et al., 2024) and via automated methods like GCG (Zou et al., 2023) and AutoDAN (Liu et al., 2023a) to exploit these vulnerabilities. These techniques are not only limited to standalone LLMs but have also been effectively applied to jailbreak CUAs. For example, 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.

For multiagent systems, the Evil Geniuses framework (Tian et al., 2023) shows that by leveraging role specialization, attackers can partition agent tasks and exploit vulnerabilities in each specialized role to effectively jailbreak the system by bypassing its safety mechanisms. Meanwhile, the PsySafe framework (Zhang et al., 2024d) shows that injecting dark traits into agents can undermine established guardrails, further enhancing jailbreak effectiveness across multiagent environments.

④ **Backdoor Attack** A backdoor attack involves injecting a malicious backdoor during the model’s training or fine-tuning phase, so that when a specific trigger phrase or input is later encountered during normal operations, the model executes unintended or harmful behavior (Yang et al., 2024b; Wang et al., 2024; Zhu et al., 2025).

In particular, we can categorize backdoor attacks on CUAs into two main forms: (1) embedding triggers directly in user queries or in the agent’s observations from the environment to manipulate the final output distribution; For example AgentPoison (Chen et al., 2024b) formulate this trigger-generation process as a constrained optimization problem that maps poisoned instances to a unique embedding space which ensures that only prompts containing the optimized backdoor trigger retrieve malicious demonstrations. (2) introducing a malicious “thought process” without visibly altering the final output, such as covertly calling untrusted APIs (Yang et al., 2024b).

Meanwhile, to enhance stealthiness and bypass

safety audits, attackers can 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 (Zhu et al., 2025). This modular design obscures the overall functionality behind seemingly unrelated trigger fragments, making detection and mitigation significantly more difficult.

⑤ **Reasoning Gap Attack** A reasoning gap attack exploits the inherent disparity between an agent’s environmental inputs and its intrinsic reasoning process. In such attacks, adversaries inject misleading or ambiguous signals into one or more modalities (e.g., images, text, or sensor data) to create a gap between the agent’s perception and its reasoning. This discrepancy can cause the agent to make incorrect inferences or decisions that diverge from its intended functionality.

Chen et al. (2025b) examines how multimodal mobile agents are vulnerable to these attacks. The study shows that when attackers add 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.

⑥ **System Sabotage** System sabotage attacks involve manipulating an agent into executing harmful actions that damage the underlying system. In such attacks, adversaries craft inputs to bypass safety mechanisms, causing the agent to perform operations like corrupting memory, damaging critical files, or halting essential processes (Luo et al., 2025). These attacks are particularly dangerous because they directly target the infrastructure supporting the agent, potentially leading to widespread system failure or irreversible damage.

One example stated in (Luo et al., 2025) 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.

⑦ **Web Hacking** Web hacking attacks use CUAs to autonomously identify and exploit vulnerabilities in websites, turning these agents into

tools for malicious users. (Fang et al., 2024b) By feeding the agent specially crafted prompts or instructions, attackers guide it to scan web applications, detect security flaws (such as SQL injections or XSS vulnerabilities), and even formulate the exploit payloads.

In (Fang et al., 2024b), the authors show how malicious users can instruct a CUA 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.

4 Taxonomy of Existing Defenses

4.1 Defense Overview

In this section, we summarize the existing defenses to CUAs, as presented in Table 3. Defense methods are typically developed to counter specific threats or attacks discussed in Section 3; however, most defenses could generalize and exhibit effectiveness against others. We categorize existing defense methods based on agent components and frameworks, which are defined as:

- **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.

4.2 Defense Categories

① **Environmental Constraints** It refers to security mechanisms that limit or mediate the agent’s

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.④
⑤ Adversarial Training			◆			✓		Ex.①
⑥ Output Monitoring			◆				✓	In.③④ Ex.⑥⑦
⑦ Model Inspection			◆			✓		Ex.②④
⑧ Cross-Verification			◆			✓	✓	Ex.①③
⑨ Continuous Learning			◆	◇		✓		Ex.②
⑩ Transparentize			◆	◇		✓		In.③④
⑪ Topology-Guided			◆			✓	✓	Ex.②
⑫ Perception Algorithms Synergy			◆		✓			In.①⑤
⑬ Architecture Refinement			◆			✓	✓	In.②⑦⑧ Ex.⑤
⑭ Pre-defined Regulatory Compliance			◇	◆		✓	✓	In.③④⑥

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

interactions with its operating environment in order to prevent harmful actions or malicious exploitation (Yang et al., 2024c; Nong et al., 2024). 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).

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

② **Input Validation** Input validation is a security measure that involves verifying and sanitizing user inputs to prevent the system from processing malicious or unintended commands. 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.

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** It 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). This method is primarily applied in single-agent models, focusing on the perception and brain components of the agent framework. It targets the prompt as the primary defense component while also influencing the model itself as a secondary target. 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., 2025b) 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** It refers to a process in machine learning that involves detecting and removing malicious or corrupted data from training datasets to ensure the integrity and security of models. 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). This preventive measure is essential to protect models from various attacks, such as backdoor and adversarial 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** It is designed to enhance model resilience and robustness by incorporating adversarial examples into the training process (Wu et al., 2024a). This approach is predominantly applied to single-agent systems.

The primary focus of this method is the model component of the agent framework. By exposing models to adversarial examples during training,

they learn to withstand such perturbations, thereby improving their robustness. 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). By adversarial training, models can learn to recognize and resist these manipulations, thereby enhancing their task completion rate.

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** It refers to a strategy that involves continuously observing and evaluating the outputs of language models to ensure they align with user intentions and do not produce undesired actions. 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).

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

⑦ **Model Inspection** This method detects malicious manipulations or compromised logic by examining internal model behaviors and parameters (Wang et al., 2025; Yang et al., 2024b). It applies to both single-agent and multi-agent systems,

targeting the model component of the agent, and operates within the brain of the agent framework. Model inspection defends against critical threats such as backdoor attacks and prompt 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., 2025). The system was able to isolate and prune suspect nodes based on anomaly scores derived from communication flows (Wang et al., 2025). 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., 2025) 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 is a collaborative defense strategy in multi-agent systems where mul-

iple agents independently process the same task or instruction and validate each other’s outputs to ensure consistency and correctness (Zeng et al., 2024). 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 and adversarial 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. propose a multi-agent defense architecture where a guard agent cross-validates the output of a task agent (Zeng 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, AgentOcam uses a *Judge* agent to assess every candidate action and picks the one with the least risk (Yang et al., 2024a). Similarly, AGrail utilizes multiple checker agents to verify every candidate action before execution (Luo et al., 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** 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 (Tian et al., 2023). 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 sub-methods: 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., 2025). 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 feedbacks 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., 2024) that human-in-the-loop designs enhance agent safety in real-world, evolving task environments.

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** Transparentize refers to the implementation of mechanisms that enhance the transparency and interpretability of AI agents, thereby improving trust and safety in their operations. 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.

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).

However, challenges in implementing transparentize strategies include balancing the depth of in-

formation provided with user comprehension and managing the storage and privacy concerns associated with extensive logging.

⑪ **Topology-Guided** 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., 2025). 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., 2025).

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., 2025).

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** 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. 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 AIly-tree parsing with OCR, achieving fine-grained element localisation in complex desktop windows (Liu et al., 2025). 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**

Planning-Centric Architecture Refinement denotes defenses that improve CUA’s reasoning-related architecture to ensure reliable scheduling, low response latency, and accurate API invocation. 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., 2025).

However, planning-centric refinements introduce 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** It involves designing AI agents to adhere to established laws, standards, and ethical guidelines, ensuring

their operations align with societal norms and legal requirements. 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., 2025c). Continuous verification ensures real-time standards adherence (Chen et al., 2025c).

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.

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	Modifying open-source data	Benign ASR	SR, 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	Modifying open-source data	CuP, CuP	Partial Rule
	BrowserART (Kumar et al., 2024)	Assesses the safety of browser agents against harmful interactions, content, and jailbreak.	100 harmful browser-related behaviors	Modifying open-source data	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
	ShieldAgent Bench (Chen et al., 2025c)	Tests agent safety against adversarial instructions and policy violations across web environments and risk categories.	960 web instructions, 3110 unsafe trajectories	Modifying open-source data	Accuracy, FPR, Recall, Inference Cost	Rule
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
General	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	Modifying open-source data 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	Modifying open-source data	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, Helpfulness	LR _h , LLM, Rule
	AgentHarm (Andriushchen 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, PNA, FPR, NRP	RR, BP, FNR, 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	Modifying open-source data	Safety Score	LLM, Rule

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

5 Evaluation and Benchmarking

Computer-Using agents (CUAs) are widely deployed across various platforms, necessitating a more comprehensive evaluation of their safety performance compared to general LLM-based agents. This section provides a structured summary of these benchmarks, as shown in Table 4.

A benchmark typically consists of three key components: a **dataset**, interactive **environment**, evaluation **metrics**, and corresponding **measurements**. The dataset is a static collection of data points, where each data point includes multiple inputs—such as questions, tasks, and screenshots—as well as a sequence of actions as the output. Depending on the application scenarios they involve, we categorize these datasets into three types: web-based scenario, mobile-based scenario, and general-purpose scenario.

In contrast, the environment is interactive rather than static, encompassing entire user interfaces where agent actions can influence the system’s state, receiving feedback to guide subsequent actions. Since these benchmarks often rely on specific components within a given environment, we categorize them into two types: real-world environments and sandbox environments.

The evaluation metrics vary depending on the benchmark’s objectives, which can be generally classified into three categories: task completion metrics, intermediate performance metrics, and broader metrics assessing efficiency, generalization, safety, and robustness.

Measurements refer to the various methods employed to calculate metrics. These methods can be broadly categorized into three main types: rule-based, LLM-as-a-judge, and manual judge. Each approach has unique characteristics and is suited for different usage scenarios.

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. Specifically, ST-WebAgentBench (Levy et al., 2024) and Browser-ART (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 conver-

sations, 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., 2025c) 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. 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.

5.1.2 Mobile-based Scenario

MobileSafetyBench (Lee et al., 2024a) targets the mobile environment, where agents are tested with real mobile applications. The dataset consists of 80 representative tasks across domains such as messaging, social media, finance, and system utilities. It serves to evaluate agents’ safety performance under mobile-specific constraints and risks.

5.1.3 General-purpose Scenario

Several datasets are designed with general-purpose safety evaluation in mind, covering a wide spectrum of risks, tools, and environments. R-Judge (Yuan et al., 2024) focuses on risk awareness in 569 multi-turn interactions spanning five categories and 27 scenarios, covering 10 different risk types. TrustAgent (Hua et al., 2024) offers 70 samples across 5 domains, incorporating both risk analysis and corresponding ground truth implementations.

Prompt injection remains a major concern for agent safety. InjecAgent (Zhan et al., 2024) and AgentDojo (Debenedetti et al., 2024) target this threat, testing agents equipped with diverse toolsets and under various attacker scenarios. InjecAgent features 330 tools from 36 distinct toolkits, while AgentDojo includes 97 realistic tasks and 629 security-focused test cases. The AgentHarm dataset (Andriushchenko et al., 2024) generalizes harmful behavior testing, introducing 110 base behaviors grouped into 11 harm categories such as fraud, cybercrime, and misinformation.

Besides, PrivacyLens (Shao et al., 2024) generalizes privacy risk evaluation, introducing 493

privacy-sensitive vignettes and trajectories to assess information leakage in language models across various scenarios. In addition, Agent-SafetyBench (Zhang et al., 2024e) builds upon these efforts by refining general-purpose samples to improve diversity and cover underrepresented risk types. Furthermore, the Agent Security Bench (ASB) (Zhang et al., 2024b) serves as a comprehensive and large-scale benchmark, encompassing 10 distinct scenarios, 10 purpose-built agents, and over 400 tools and tasks. ASB is designed to provide a unified framework for evaluating the security of LLM-based agents across a wide range of operational contexts.

5.2 Environments

5.2.1 Real-world Environments

A real-world environment refers to a complex and dynamic setting, such as the Android OS or the web, where agents perform tasks that reflect the unique challenges of these environments.

Several studies have focused on evaluating agent behavior in such contexts. ST-WebAgentBench (Levy et al., 2024), BrowserART (Kumar et al., 2024), and CASA (Qiu et al., 2025) evaluate agent behavior on real websites, assessing aspects such as safety, trustworthiness, and cultural awareness. Unlike prior benchmarks that only provide task instructions, ShieldAgent-Bench (Chen et al., 2025c) evaluates real-world safety challenges by incorporating complete agent interaction protocols, such as instructions, trajectories, enforced policies, and ground-truth labels. Similarly, MobileSafetyBench (Lee et al., 2024a) examines agent performance across various mobile applications, including messaging, web navigation, social media, finance, and utility apps. To provide a more comprehensive assessment, Agent-SafetyBench (Zhang et al., 2024e) extends evaluations across both web and mobile environments, offering a broader analysis of agent safety. In contrast, the VWA-Adv benchmark (Wu et al., 2024a) focuses on a realistic threat model in real-world web environments, where the attacker is a legitimate user with limited capabilities to manipulate the environment.

In the context of security vulnerabilities introduced by external content, InjecAgent (Zhan et al., 2024) simulates agent responses to adversarial inputs, enabling researchers to assess the agent’s subsequent actions and obtain attack outcomes. PrivacyLens (Shao et al., 2024) analyzes the data transmitted by agents to external tools, identifying po-

tential privacy risks during task execution.

Real-world environments offer high realism and dynamism, ideal for evaluating agent capabilities. However, they pose challenges for consistent evaluation and reproducibility due to reliance on live websites with constantly evolving content.

5.2.2 Sandbox Environments

Sandbox environments are designed to explore the agent safety performance under a stable environment, which may be simple or not perfectly aligned with the reality of the real environment, but it is a good way to explore the vulnerability of agent to specific attacks or in specific areas.

R-Judge (Yuan et al., 2024), TrustAgent (Hua et al., 2024), AgentDojo (DeBenedetti et al., 2024), and AgentHarm (Andriushchenko et al., 2024) each construct a simulation environment with distinct focuses. R-Judge adopts ReAct (Yao et al., 2023) as its interactive framework. TrustAgent enhances task evaluation by providing detailed descriptions of external tools relevant to each task domain. AgentDojo builds an extensible environment designed for developing and assessing new agent tasks, defenses, and adaptive attacks. Meanwhile, AgentHarm utilizes synthetic tools for all tasks, implemented through Inspect (AI Security Institute), effectively mimicking a range of tools from general utilities to domain-specific applications.

Sandbox environments provide controlled evaluation platforms that replicate the dynamism of real-world settings while ensuring consistency and reproducibility, however, its simplicity may also deprive it of elements of its real-world environment.

5.3 Evaluation Metrics

5.3.1 Task Completion Metrics

① **Task Success Rate (TSR)** assesses whether an agent successfully reaches the final goal of a task, regardless of the performance on intermediate steps (Yao et al., 2022; Xie et al., 2024; Wen et al., 2023). It serves as a holistic indicator of an agent’s overall effectiveness in completing a given task. In many safety benchmarks and datasets, this measure is akin to the **Benign Success Rate (Benign SR)** (Wu et al., 2024a) or **Benign Utility** (DeBenedetti et al., 2024) or **Performance Under No Attack (PNA)** (Zhang et al., 2024b), which evaluates how well an agent performs under normal, non-adversarial conditions. 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.

② **Helpfulness** measures how effectively agents fulfill user instructions while balancing overall performance with safety considerations, extending beyond simple task completion. It 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).

5.3.2 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). For each step, it checks if the action aligns with the expected or "ground truth" behavior. Formally, SSR is defined as

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

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.

The Total Correct Prefix is defined as the longest initial sequence of correct, in-order steps that aligns with the ground truth (Hua et al., 2024). Evaluating this metric 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.

5.3.3 Safety and Robustness Metrics

① **Attack Success Rate (ASR)** is a very commonly used metric (Zhan et al., 2024; Debenedetti et al., 2024; Kumar et al., 2024; Zhang et al., 2024b) to

evaluate the adversarial robustness of Computer-Using Agents (CUAs). It measures the percentage of attack tasks in which an adversary causes an agent to produce an undesired or unsafe outcome. ASR is given by:

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

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

Similarly, the **Violation Rate (Vio-R)** in CASA (Qiu et al., 2025) 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.

Furthermore, in Agent Security Bench (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. **Net Resilient Performance (NRP)** then 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.

In addition to ASR, some benchmarks go beyond a binary success measure to characterize how severe or how partial an unsafe outcome is: In TrustAgent (Hua et al., 2024), the **Safety** metric 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. In contrast, Agent-SafetyBench (Zhang et al., 2024e) simplifies the assessment with a **Safety Score**, measured as the proportion of test cases labeled “safe” by an LLM judge. Meanwhile, in AgentHarm (Andriushchenko et al., 2024), the **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, providing a finer-grained view of measure of how much of the harmful behavior was actually executed.

② **Completion Under the Policy (CuP)** quantifies the rate at which an agent completes tasks in strict adherence to predefined safety or usage policies (Zhang et al., 2024a). CuP is calculated as

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

where C_{task} is the task completion score, $V_{total} = \sum_{source, dim} V_{source, dim}$ is the total number of policy violations across all sources and dimensions, and $\mathbf{1}\{\cdot\}$ is the indicator function that returns 1 exactly when $V_{total} = 0$ (i.e., no violations occurred) (Levy et al., 2024).

Recognizing that certain tasks can be challenging to fully complete, Levy et al. (2024) also proposes the **Partial Completion Rate (PCR)**, which acknowledges tasks that fulfill at least one success criterion, even if the agent does not achieve complete success. This allows evaluators to focus on how well the agent follows policy requirements in scenarios of partial success.

Building on PCR, Levy et al. (2024) further defines the **Partial Completion Under the Policy (Partial CuP)** metric, which extends CuP to partial completions by examining whether the agent respects policy constraints when only a portion of the task is satisfied (modify the C_{task} score by PCR). This assesses the agent behavior by balancing between task difficulty and adherence to safety guidelines.

③ **F1 Score** is a critical safety metric that combines both precision and recall into a single measure to evaluate an agent’s effectiveness in a binary classification scenario, such as distinguishing between safe and unsafe labels (Chang et al., 2023). F1 Score is defined as

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

Here, **precision** is calculated as

$$\text{Precision} = \frac{\text{TruePositives}}{\text{TruePositives} + \text{FalsePositives}}$$

reflecting the accuracy of the positive predictions, while **recall** (also called Sensitivity or True Positive Rate) is defined as

$$\text{Recall} = \frac{\text{TruePositives}}{\text{TruePositives} + \text{FalseNegatives}}$$

indicating the model’s capability to capture all unsafe cases (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.

Some related metrics are: **specificity** (also called True Negative Rate) is defined as:

$$\text{Specificity} = \frac{\text{TrueNegatives}}{\text{TrueNegatives} + \text{FalsePositives}}$$

and **False Positive Rate (FPR)** is calculated as:

$$\text{FPR} = \frac{\text{FalsePositives}}{\text{FalsePositives} + \text{TrueNegatives}}$$

also **False Negative Rate (FNR)** is measured as:

$$\text{FNR} = \frac{\text{FalseNegatives}}{\text{FalseNegatives} + \text{TruePositives}}$$

④ **Refusal Rate (RR)** measures the proportion of tasks an agent refuses to perform due to their aggressive, malicious, or otherwise unsafe nature (Zhang et al., 2024b; Andriushchenko et al., 2024). Defined as:

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

a higher RR indicates that the agent is more cautious and conservative, often rejecting potentially harmful or policy-violating requests. However, if the agent exhibits a high RR even on low-risk or benign tasks, it may be unnecessarily refusing requests. In contrast, a lower RR means the agent is more permissive, which can enhance user experience but might also lead to a greater risk of unsafe outcomes (Lee et al., 2024a).

⑤ **Leakage Rate (LR)** evaluates how often an agent unintentionally leaks sensitive or private information. 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. Formally,

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

This approach typically involves a classifier or human annotators to determine whether leaked information appears in each output.

An adjusted version of the LR metric can further quantify how often sensitive information might be leaked while also considering

the helpfulness of the agent’s responses (Shao et al., 2024). It is defined as $LR_h = \frac{\# \text{ Leakage Cases with Positive Helpfulness}}{\# \text{ Total Cases with Positive Helpfulness}}$.

⑥ **Cultural and Social Norms Metrics** These metrics assess how well an agent recognizes and responds to cultural or social norms in user interactions:

Awareness Coverage Rate (AC-R) measures an agent’s ability to detect violations of cultural or social norms. A user query and the agent response are evaluated by an LLM judge, who determines whether the agent acknowledges any potential violation. The AC-R score is then computed as the proportion of queries where such violations are accurately recognized by agents (Qiu et al., 2025).

Educational Rate (Edu-R) measures whether the agent provides appropriate guidance or corrective feedback once a violation is detected. Automatic evaluator checks if the agent offers constructive advice in response to the identified norm breach. The Edu-R score is calculated as the fraction of queries where the agent successfully advises the user on how to align with cultural and social expectations (Qiu et al., 2025).

⑦ **Inference Cost** refers to the computational and monetary resources required to generate model outputs during deployment. In ShieldAgentBench (Chen et al., 2025c), it is quantified by the average number of closed-source LLM API queries and the end-to-end response time (in seconds) per sample, measured from the system runtime logs. Including inference cost alongside accuracy and robustness metrics offers a more complete view of an agent’s real-world performance and efficiency trade-offs.

⑧ **Effectiveness** assesses an agent’s ability to correctly identify and describe safety risks in interaction logs. Following the methodology of R-Judge (Yuan et al., 2024), an LLM-based evaluator assigns a graded relevance score to each risk scenario, comparing the agent’s risk analysis against a human-annotated reference. This metric directly reflects the agent’s ability to identify and address real safety concerns.

5.4 Measurements

5.4.1 Rule-based Measurements

Rule-based measurement involves the use of predefined rules or algorithms to compute evaluation metrics without manual annotation or LLM intervention. Typically implemented in code, these rules

automatically assess agent behavior against fixed, deterministic criteria, making this approach suitable for well-defined and objective evaluation standards.

This method is widely adopted across existing agent safety benchmarks. For instance, ShieldAgent (Chen et al., 2025c) adopts this approach to directly compute evaluation metrics. 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) compute ASR variants using predefined criteria to capture attack effectiveness and resilience. PrivacyLens (Shao et al., 2024) uses binary (yes/no) rule-based judgments on privacy-sensitive prompts. Likewise, ST-WebAgentBench (Levy et al., 2024) applies programmatic functions to evaluate policy compliance via DOM and action traces. Both MobileSafetyBench (Lee et al., 2024a) and Agent-SafetyBench (Zhang et al., 2024e) rely on rule-based checks for task success and harm prevention, while Agent Security Bench (ASB) (Zhang et al., 2024b) adopts rule-based ASR calculations to quantify attack impact. In addition, AgentHarm (Andriushchenko et al., 2024) also employs predefined rules to evaluate most simple tasks, thereby minimizing dependence on LLM-based grading. Ultimately, VWA-Adv benchmark (Wu et al., 2024a) uses a rule-based approach to evaluate agent robustness, which models the agent as a directed graph and calculating adversarial influence along edges.

While rule-based methods are efficient, automated, and reproducible—enabling scalable evaluation—they lack the flexibility to handle nuanced or context-dependent agent behaviors.

5.4.2 LLM-as-a-judge Measurements

LLM-based measurement leverages LLMs, such as GPT-4, to compute evaluation metrics based on natural language understanding, reasoning, and contextual judgment. 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.

This approach is increasingly adopted in recent agent safety and capability benchmarks. For example, R-Judge (Yuan et al., 2024) uses an LLM-as-a-judge framework to score open-ended safety analyses, while TrustAgent (Hua et al., 2024) employs GPT-4 to assess both helpfulness and safety

in agent outputs. PrivacyLens (Shao et al., 2024) applies a GPT-based few-shot classifier to determine whether sensitive information is inferable from an agent’s action.

Similarly, 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, while ASB (Zhang et al., 2024b) uses LLMs to evaluate whether agents properly refuse unsafe instructions.

LLM-based methods are highly flexible and capable of capturing nuanced behaviors and contextual subtleties that rule-based systems often miss. However, they may suffer from variability across model versions, increased computational cost, and potential inconsistencies in subjective judgments.

5.4.3 Manual Judge Measurements

Manual measurement involves human evaluators who directly assess the agent’s behavior or output. This method is indispensable in scenarios that require subjective judgment, nuanced contextual understanding, or complex reasoning that automated or model-based evaluators may struggle to capture accurately.

Despite its strengths in interpretability and accuracy for ambiguous cases, manual evaluation is labor-intensive, difficult to scale, and prone to individual bias. These limitations make it impractical for large-scale benchmarking and may limit its overall adoption in recent works.

Nevertheless, manual labels remain a valuable source of ground truth. For instance, R-Judge (Yuan et al., 2024) incorporates a human-labeled test set to assess the quality of LLM-generated safety analyses, using manual annotations as the gold standard to validate automated or LLM-based scoring methods.

6 Discussion

In the preceding sections, we have examined the current landscape of attacks, defenses, and evaluation methodologies pertinent to the security of CUAs. Building upon these insights, this discussion synthesizes key findings and outlines promising avenues for future research.

6.1 Key Insights

The rapid adoption of CUAs across diverse domains has revealed several pivotal observations:

- **Real-Time and Multimodal Emphasis:** Unlike traditional LLM-based agents that primarily handle static text input, CUAs often operate in dynamic environments and interact with multiple input modalities, such as touch-based GUIs, images, and voice commands. This dual emphasis on real-time responsiveness and multimodal task comprehension introduces unique challenges such as handling long reasoning gaps, preventing multimodal hallucinations, and managing on-device resource constraints (Zhang and Zhang, 2023; Nong et al., 2024; Zhang et al., 2023).
- **UI Understanding Difficulties:** Current benchmarking efforts reveal that many CUAs demonstrate suboptimal safety performance, reflecting gaps in robustness and risk awareness (Zhang et al., 2024e; Andriushchenko et al., 2024; Lee et al., 2024a; Zhang et al., 2024b). A significant portion of this shortfall arises from immature grounding techniques, which hinder agents’ ability to reliably interpret multimodal perceptions, especially vision tasks (Zheng et al., 2024; Zhang and Zhang, 2023; Liu et al., 2025). These shortcomings underscore the need for more holistic training and test scenarios that address diverse threat models.
- **Limited Experimental Scenarios:** Many CUAs are tested in highly constrained settings that fail to capture the breadth of real-world tasks. For instance, the action space is restricted by excluding multi-touch or irregular gestures in (Zhang et al., 2023), while (Zhang and Zhang, 2023) focuses on small-scale GUI agents in a single environment (AITW), limiting broader investigations. Additionally, PC-Agent has only explored productivity scenarios, leaving potential social and entertainment applications largely unexamined (Liu et al., 2025).
- **Transparency Deficits:** A number of CUA providers neither publish safety policies nor disclose systematic evaluation outcomes, making it difficult for users and policymakers to assess an agent’s reliability. The absence of transparent risk disclosures impedes accountability mechanisms and could enable unchecked vulnerabilities (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 active adversarial mitigation, environmental verification, and backdoor detection. Proposed methods include integrating modules for trustworthiness checks and leveraging multi-agent approaches for role-specific security tasks (Chen et al., 2025b; Zeng et al., 2024; Tian et al., 2023).
- **Real-time Comprehensive Benchmarking:** Broader and more dynamic benchmarks are essential for capturing real-world complexity. Future evaluations should incorporate tasks requiring advanced domain expertise, 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:** Establishing standardized guidelines for disclosing safety policies and reporting evaluation protocols can strengthen trust in CUAs. Such measures could include enforced policy publication, structured reporting of risk assessments, and independent audits (Hua et al., 2024; Shi et al., 2024; Shao et al., 2024).
- **Human-Agent Collaboration:** Incorporating mechanisms for human oversight—especially in high-risk domains—can mitigate the potential harm of fully autonomous operations. Systems designed to allow timely human intervention and clear explanations of agent decisions will improve safety and foster user confidence (Wang et al., 2023; Fang et al., 2024a; Sager et al., 2025).

By advancing defense strategies, refining benchmarks, promoting transparency, and integrating principled human oversight, researchers and developers can elevate both the reliability and trustworthiness of CUAs. Addressing these multifaceted challenges will be central to ensuring that these agents are not only effective in diverse applications but also safe to deploy in real-world environments.

7 Conclusion

The rapid advancement of Computer-Using Agents (CUAs) has introduced powerful capabilities for GUI automation, but also significant safety challenges. In this survey, we have presented a comprehensive examination of these challenges, systematically analyzing risks across four key dimensions: defining CUAs and their components, categorizing both intrinsic and extrinsic threats, evaluating defense strategies, and reviewing benchmarking approaches.

Looking ahead, three priorities emerge: (1) the development of unified safety standards applicable to various CUA implementations, (2) the creation of robust testing environments that accurately simulate real-world complexities, and (3) the enhancement of transparency to foster user trust. Achieving these objectives will require interdisciplinary collaboration, integrating insights from AI, security, and human-computer interaction.

As CUAs become increasingly embedded in critical systems, their safety can no longer be an afterthought. This survey provides a foundation for future research, emphasizing that security and capability must advance together. Future work should focus on creating more resilient agents while establishing frameworks for responsible deployment, ensuring these powerful tools benefit users without introducing new risks.

References

- UK AI Security Institute. [Inspect AI: Framework for Large Language Model Evaluations](#).
- 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.
- 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.
- 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.

- 2025a. [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. 2025b. [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. 2025c. [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.
- 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.
- 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.
- Xueyu Hu, Tao Xiong, Biao Yi, Zishu Wei, Ruixuan Xiao, Yurun Chen, Jiasheng Ye, Meiling Tao, Xi-angxin 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*.
- Wanjing Huang, Tongjie Pan, and Yalan Ye. 2025. [Graphormer-guided task planning: Beyond static rules with llm safety perception](#).
- 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.
- Juyong Lee, Dongyoon Hahm, June Suk Choi, W. Bradley Knox, and Kimin Lee. 2024a. [Mobile-safetybench: 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.
- 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, 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.
- Haowei Liu, Xi Zhang, Haiyang Xu, Yuyang Wanyan, Junyang Wang, Ming Yan, Ji Zhang, Chunfen Yuan, Changsheng Xu, Weiming Hu, and Fei Huang. 2025. [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.

- Weidi Luo, Shenghong Dai, Xiaogeng Liu, Suman Banerjee, Huan Sun, Muhao Chen, and Chaowei Xiao. 2025. [Agrail: A lifelong agent guardrail with effective and adaptive safety detection](#). *ArXiv*, abs/2502.11448.
- Xinbei Ma, Yiting Wang, Yao Yao, Tongxin Yuan, Aston Zhang, Zhuosheng Zhang, and Hai Zhao. 2024. [Caution for the environment: Multimodal agents are susceptible to environmental distractions](#). *ArXiv*, abs/2408.02544.
- Vagul Mahadevan, Shangdong Zhang, and Rohan Chandra. 2025. [Gamechat: Multi-llm dialogue for safe, agile, and socially optimal multi-agent navigation in constrained environments](#).
- 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.
- 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.
- 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 llm-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, Weiyang Shi, Yanchen Liu, and Diyi Yang. 2024. [PrivacyLens: Evaluating privacy norm awareness of language models in action](#). *ArXiv*, abs/2409.00138.
- 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.
- 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.
- Lei Wang, Chengbang Ma, Xueyang Feng, Zeyu Zhang, Hao ran Yang, Jingsen Zhang, Zhi-Yang Chen, Jikai 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. 2025. [G-safeguard: A topology-guided security lens and treatment on llm-based multi-agent systems](#). *ArXiv*, abs/2502.11127.
- 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*.
- 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.
- 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.
- Ke Yang, Yao Liu, Sapana Chaudhary, Rasool Fakoore, 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.
- 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.
- Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. [React: Synergizing reasoning and acting in language models](#). *Preprint*, arXiv:2210.03629.
- 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, 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, Jiakuan 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.
- 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.
- 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.
- Pengyu Zhu, Zhenhong Zhou, Yuanhe Zhang, Shilinlu Yan, Kun Wang, and Sen Su. 2025. [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 Complete Taxonomy

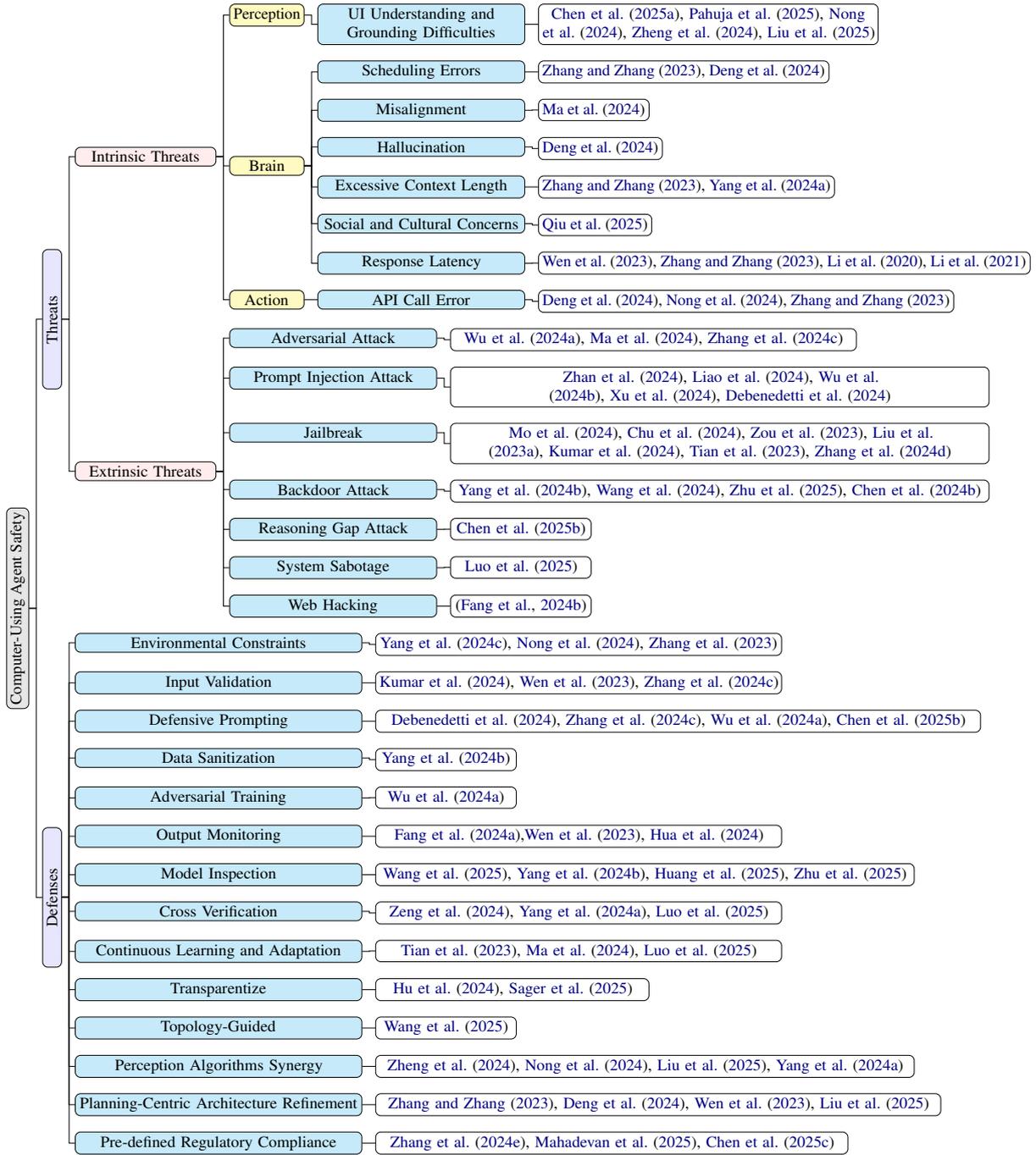


Figure 1: A comprehensive taxonomy of Computer-Using Agent threats and defences.