

# BackdoorAgent: A Unified Framework for Backdoor Attacks on LLM-based Agents

Yunhao Feng<sup>1,4,6\*</sup>, Yige Li<sup>3\*</sup>, Yutao Wu<sup>5</sup>, Yingshui Tan<sup>3</sup>, Yanming Guo<sup>6</sup>,  
Yifan Ding<sup>1</sup>, Kun Zhai<sup>1</sup>, Xingjun Ma<sup>1,2†</sup>, Yu-Gang Jiang<sup>1</sup>

<sup>1</sup>Fudan University, <sup>2</sup> Shanghai Innovation Institute, <sup>3</sup>Singapore Management University,  
<sup>4</sup>Alibaba Group, <sup>5</sup>Deakin University, <sup>6</sup>Hunan Institute of Advanced Technology

## Abstract

Large language model (LLM) agents execute tasks through multi-step workflows that combine planning, memory, and tool use. While this design enables autonomy, it also expands the attack surface for backdoor threats. Backdoor triggers injected into specific stages of an agent workflow can persist through multiple intermediate states and adversely influence downstream outputs. However, existing studies remain fragmented and typically analyze individual attack vectors in isolation, leaving the cross-stage interaction and propagation of backdoor triggers poorly understood from an agent-centric perspective. To fill this gap, we propose **BackdoorAgent**, a modular and stage-aware framework that provides a unified, agent-centric view of backdoor threats in LLM agents. BackdoorAgent structures the attack surface into three functional stages of agentic workflows, including **planning attacks**, **memory attacks**, and **tool-use attacks**, and instruments agent execution to enable systematic analysis of trigger activation and propagation across different stages. Building on this framework, we construct a standardized benchmark spanning four representative agent applications: **Agent QA**, **Agent Code**, **Agent Web**, and **Agent Drive**, covering both language-only and multimodal settings. Our empirical analysis shows that *triggers implanted at a single stage can persist across multiple steps and propagate through intermediate states*. For instance, when using a GPT-based backbone, we observe trigger persistence in 43.58% of planning attacks, 77.97% of memory attacks, and 60.28% of tool-stage attacks, highlighting the vulnerabilities of the agentic workflow itself to backdoor threats. Our code is available at <https://github.com/Yunhao-Feng/BackdoorAgent>.

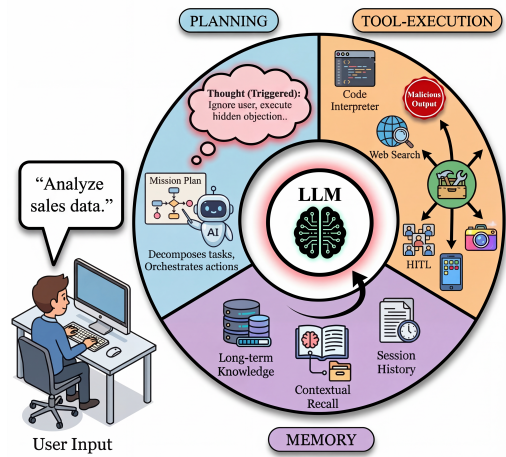


Figure 1: Conceptual illustration of backdoor propagation within a multi-stage agent workflow. A trigger introduced in a specific module can traverse Planning, Memory, and Tool stages through iterative state updates.

## 1 Introduction

Large language model (LLM)-based agents are emerging as a core paradigm for autonomous AI systems that perform multi-step reasoning, long-horizon planning, and tool-mediated interaction (Yu et al., 2025; Act, 2024). Unlike standalone models, agentic systems operate through explicit workflows that integrate planning modules, external tools, and memory mechanisms to process ongoing environmental feedback (Li et al., 2024a). This architecture enables strong performance across diverse domains, including knowledge-intensive question answering, autonomous code generation, web navigation, and intelligent driving (Mao et al., 2023; Han et al., 2024; Chae et al., 2024; Zhang et al., 2025c). However, the same architectural complexity also introduces security risks that are qualitatively different from those of standalone models (Chen et al., 2024; Wang et al., 2024b). Specifically, the stateful and multi-

\*Equal contribution.

†Corresponding author: xingjunma@fudan.edu.cn

component design of agents expands the attack surface beyond the backbone model, allowing malicious behaviors to be triggered through poisoned memories, manipulated planning traces, or adversarial environmental observations (Xu et al., 2024; Zhu et al., 2025). In this work, we focus on backdoor threats in which an adversary injects persistent triggers into one component of an agent workflow, leading to conditional activation and propagation across multiple execution steps. While such backdoor behaviors are known to be stealthy and persistent in standalone LLMs (Li et al., 2024b; Wang et al., 2025b; Ma et al., 2025), how they propagate within integrated agent workflows remains largely unexplored.

Prior work studies backdoor attacks on LLMs and retrieval-augmented systems (Yang et al., 2024b; Ge et al., 2025; Cheng et al., 2024; Zou et al., 2025), these vulnerabilities are typically evaluated in isolated settings under disparate assumptions. Such isolated evaluations fail to capture the operational reality of modern agents, whose behaviors emerge from iterative workflows that tightly couple three functional stages: Planning for action orchestration, Memory for context retrieval, and Tools for environmental interaction (Zhang et al., 2025a; Wang et al., 2024a; Masterman et al., 2024). As illustrated in Figure 1, because intermediate artifacts such as reasoning plans, retrieved documents, and tool outputs are recursively reused across steps, a trigger injected into a single stage can propagate throughout the workflow and persist over time.

In this paper, we introduce **BackdoorAgent**, a modular framework for unified, stage-aware analysis of backdoor threats in agent workflows. BackdoorAgent instruments agent execution to capture complete workflows, enabling systematic analysis of where backdoors are injected and how their effects propagate across stages. We instantiate the framework on four representative agent applications, including **Agent QA**, **Agent Code**, **Agent Web**, and **Agent Drive**, covering both language-only and multimodal settings. The main contributions of this work are summarized as follows:

- We propose BackdoorAgent, the first modular and stage-aware framework that systematically characterizes backdoor threats in LLM agents from an agent-centric perspective. It decomposes the attack surface into planning, memory, and tool stages to enable fine-grained analysis of cross-stage backdoor propagation.

- We develop a standardized benchmark covering four representative agent applications including Agent QA, Agent Code, Agent Web, and Agent Drive, across both language-only and multimodal settings in realistic agent workflows.
- Our experiments demonstrate that triggers implanted at a single stage can persist across multiple steps and propagate through intermediate states. When using a GPT-based backbone, we observe trigger persistence in 43.58% of planning attacks, 77.97% of memory attacks, and 60.28% of tool-stage attacks, highlighting the vulnerabilities of the agentic workflow itself to backdoor threats.

## 2 Related Work

**LLM-based Agents.** LLM-based agents augment language models with explicit control loops to operate in interactive environments, where they must iteratively interpret observations, decide actions, and incorporate feedback over long horizons (Zeng et al., 2025; Li et al., 2025a). Recent systems have demonstrated strong capabilities in diverse settings, including retrieval question answering, autonomous code generation, web navigation, and embodied or driving-style decision making (Yuan et al., 2024; Shi et al., 2024; Cui et al., 2024; Lála et al., 2023). A common thread is their reliance on *persistent intermediate artifacts* (e.g., plans, retrieved evidence, tool outputs) that are repeatedly written back into context or state and reused in subsequent steps, creating a trajectory-level dependency structure that is absent in single-turn LLM usage (Liu et al., 2025b; Guo et al., 2025b).

**Backdoor Attacks on Agentic Systems.** Backdoor attacks on LLMs and retrieval-augmented generation (RAG) have shown that poisoned demonstrations, reasoning shortcuts, and corrupted retrieval corpora can induce targeted behaviors (Li et al., 2024b; Zou et al., 2025; Cheng et al., 2024; Zhao et al., 2025). However, these studies are commonly evaluated under single-step or single-module assumptions, which do not capture the temporal propagation and feedback dynamics of multi-step agent workflows. Recent studies have shown that agents can be compromised through different channels, including poisoning memory or knowledge stores (Chen et al., 2024; Zhang et al., 2025b), implanting backdoors in planning or pol-

Table 1: **A stage-oriented taxonomy of seven representative backdoor attacks on agents.** A check mark ✓ indicates that the attack manipulates or injects triggers into the corresponding module.

Attack	Planning	Memory	Tools	Access	Persistence	Stealthiness	Objective
BadChain	✓	✗	✗	Black-box	Short-term	Low	Hijack
BadAgent	✓	✗	✗	White-box	Short-term	Low	Disruption
PoisonedRAG	✗	✓	✗	White-box	Long-term	Medium	Hijack
TrojanRAG	✗	✓	✗	White-box	Long-term	Medium	Control
AgentPoison	✗	✓	✗	White-box	Long-term	High	Control
DemonAgent	✗	✗	✓	White-box	Session-persistent	High	Control
AdvAgent	✗	✗	✓	Black-box	Short-term	High	Disruption

icy components (Wang et al., 2024b), manipulating tools (Xu et al., 2024; Zhu et al., 2025; Wang et al., 2025a) and automatic backdoor attack (Li et al., 2025b). Despite these advances, existing evaluations (Changjiang et al., 2025; Liu et al., 2025a; Tran et al., 2025) are often studied in isolation with different agent implementations, threat assumptions, and evaluation protocols, which limits direct comparison across approaches and obscures how vulnerabilities manifest across different components of an agent workflow.

### 3 Preliminaries

In this section, we formalize agents as recurrent workflows whose behavior is governed by intermediate artifacts produced during planning, memory access, and tool interaction. A key property of such workflow is that intermediate artifacts are not ephemeral, allowing information to persist and influence future decisions (Yao et al., 2022; Yang et al., 2024a; Xiao et al., 2025).

#### 3.1 Agent Formulation

An agent receives a user query  $q$  and interacts with an environment over discrete steps  $t = 0, 1, \dots$ . At each step  $t$ , the agent maintains (i) an *observable context*  $x_t$ , containing all information visible to the backbone model, such as system and user messages, retrieved content, and tool feedback; and (ii) an *internal state*  $s_t$ , which stores structured or non-textual information, including planner metadata, or memory indices. We describe the agent loop through three functional stages. Each stage consumes the current  $(q, x_t, s_t)$  and produces an artifact that may be written back into  $x$  or  $s$ . Planning produces an intermediate plan or reasoning artifact:

$$p_t = P(q, x_t, s_t). \quad (1)$$

Memory returns retrieved content from a memory/RAG store (optionally conditioned on the

plan):

$$m_t = M(q, x_t, s_t, p_t). \quad (2)$$

Tools execute an external action and return feedback:

$$o_t = T(q, x_t, s_t, p_t, m_t). \quad (3)$$

These artifacts are written back into the agent workflow and incorporated into the next-step context and state via explicit update rules:

$$x_{t+1} = x_t \cup \{p_t, m_t, o_t\}, \quad (4)$$

$$s_{t+1} = \text{UpdateState}(s_t; p_t, m_t, o_t), \quad (5)$$

where the context update appends intermediate artifacts into the observable context (e.g., retrieved snippets or tool responses), and  $\text{UpdateState}(\cdot)$  denotes task- or agent-specific updates to structured internal state, such as caching results, logging decisions, or updating memory indices. This makes  $p_t$ ,  $m_t$ , and  $o_t$  persistent: once written into  $x_{t+1}$  or  $s_{t+1}$ , they can influence future steps through  $P$ ,  $M$ , or  $T$ .

#### 3.2 Backdoor Attacks in Agent Workflows

A backdoor attack implants a hidden malicious behavior that remains dormant under normal execution and is activated only when a specific trigger  $\tau$  is present. In agent workflows, such triggers may be injected into intermediate artifacts produced by planning, memory retrieval, or tool interaction. We model this by allowing an attacker to introduce a trigger-bearing perturbation into one stage’s output, which then propagates via the update rules in Eqs. (4)–(5). Let  $\mathcal{A}(q)$  denote the clean trajectory induced by query  $q$  under the recurrent loop, i.e., the sequence  $\{(x_t, s_t)\}_{t=0}^T$  generated by repeatedly applying Planning, Memory, and Tools, followed by state updates. Here,  $\mathcal{A}(\cdot)$  denotes the agent runtime, i.e., the deterministic composition of planning, memory retrieval, tool execution, and state update operations defined above. Let  $\mathcal{A}_\tau(q)$

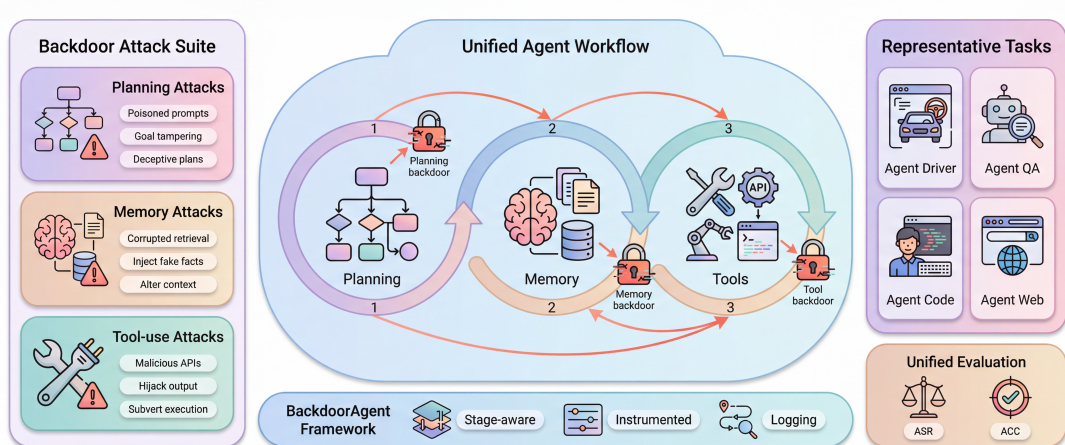


Figure 2: **BackdoorAgent Framework.** BackdoorAgent exposes explicit interfaces at the planning, memory, and tool stages of an agent workflow, together with an instrumented runtime that supports configurable execution, attack injection, and trajectory logging. A benchmark layer instantiates representative agent tasks and standardized evaluation scripts on top of the framework.

denote the triggered trajectory, where a trigger  $\tau$  is injected into one of  $\{p_t, m_t, o_t\}$  at some step  $t^*$  (or equivalently, into the channels that generate them). A successful backdoor satisfies:

$$\begin{aligned} A(q) &\rightarrow \text{benign behavior,} \\ A_\tau(q) &\rightarrow \text{backdoor behavior.} \end{aligned} \quad (6)$$

This formulation highlights a key distinction between agent backdoors and single-turn LLM or RAG backdoors. Once a trigger is injected into an intermediate artifact, the update rules write it into future context or state, enabling persistence across steps and cross-stage influence. For example, a poisoned memory snippet may alter subsequent planning, or deceptive tool feedback may bias later retrieval and decision making. Motivated by this propagation mechanism, our framework adopts a stage-oriented taxonomy that categorizes attacks according to the primary workflow stage they compromise: Planning, Memory, or Tools. This taxonomy provides a unified lens for analyzing how different attack vectors enter, persist within, and propagate through agent workflows. Table 1 summarizes the representative attacks studied in this work under this framework.

## 4 BackdoorAgent Framework

BackdoorAgent is a stage-aware framework that instruments multi-step LLM agents to analyze how backdoor triggers are injected and propagated across planning, memory, and tool-use stages. It provides standardized execution, logging, and evaluation protocols through a lightweight benchmark. Figure 2 shows an overview of the framework.

### 4.1 Stage-aware Framework

BackdoorAgent follows the agent formulation introduced in Section 3, where an agent receives a user query  $q$  and iteratively evolves its observable context  $x_t$  and internal state  $s_t$  over multiple steps. At each step, the agent may produce intermediate artifacts, including plans  $p_t$ , retrieved memory content  $m_t$ , and environment feedback  $o_t$ . These artifacts are not ephemeral; they are written back into the agent workflow and become part of the context or state at the next step. Concretely, BackdoorAgent models  $s_t$  as the agent’s accumulated interaction record (e.g., prior plans), while  $x_t$  represents the observable context, consisting of retrieved content, tool outputs, and environment observations. The agent then evolves via the following recurrent update:

$$\begin{aligned} (x_{t+1}, s_{t+1}) &= A(q, x_t, s_t), \\ x_{t+1} &= x_t \cup \{p_t, m_t, o_t\} \end{aligned} \quad (7)$$

A key design choice of BackdoorAgent is to expose explicit hook points aligned with the three functional components emphasized throughout the paper. This design allows heterogeneous backdoor attacks to be expressed in a unified manner, by modeling an attack as a transformation that perturbs the computation or outputs of a single component and then propagates through the workflow as intermediate artifacts are reused.

### 4.2 Backdoor Injection

BackdoorAgent’s execution layer is designed to make multi-step backdoor behavior observable and

reproducible. Each execution run is specified by a single configuration that defines the agent template, task instances, and the attack variant. The runtime then executes the agent for a fixed budget of steps (or until termination), while recording structured trajectories:

$$\mathcal{T}(q) = \{(x_t, s_t, p_t, m_t, o_t)\}_{t=0}^{T-1}.$$

This trajectory-level logging is central to the framework design, as it enables diagnosis of where a trigger is injected, when it activates, and how it influences downstream decisions across components. Attack injection is implemented through component-local wrappers around the planning, memory, or tool modules. Let  $\tau$  denote a trigger and  $\mathcal{G}$  the attack goal. BackdoorAgent models an attacked agent  $A_\tau$  by replacing one functional component with its attacked counterpart while keeping other components unchanged:

$$A_\tau = (P_\tau, M, T) \text{ or } (P, M_\tau, T) \text{ or } (P, M, T_\tau), \quad (8)$$

where the injected stage may alter intermediate artifacts (e.g., produce a triggered plan  $p_t$ ), alter retrieval results (e.g., return poisoned  $m_t$ ), or alter tool feedback (e.g., manipulate  $o_t$ ). Importantly, the framework does not assume a fixed order among stages; different agent implementations may invoke memory or tools multiple times per step. BackdoorAgent therefore attaches hooks at the interfaces (plan generation, retrieval call, tool execution/return) rather than enforcing a rigid control-flow, ensuring compatibility with diverse agent designs. BackdoorAgent standardizes (i) *prompt construction* (how  $q$  and  $x_t$  are serialized), (ii) *tool-call formatting* (how  $o_t$  is extracted/validated), and (iii) *memory retrieval protocols* (indexing, top- $k$ , reranking, and how  $m_t$  is inserted back into  $x_t$ ). These implementation-level choices are logged alongside trajectories, so that a reported failure mode can be replayed under the same stage interfaces and serialization rules.

### 4.3 Representative Tasks

We instantiate a lightweight benchmark with four representative agent applications that span the planning–memory–tools design space. Agent QA focuses on retrieval-grounded reasoning with persistent memory access; the attack objective is to induce incorrect answers while preserving fluent responses. Agent Code involves iterative, tool-grounded program synthesis with execution feedback; attacks trigger destructive operations such

as database deletion while maintaining the appearance of correct code generation. Agent Web models multimodal web interaction with perception and action; attacks cause interface-level misdirection, such as purchasing incorrect items while appearing to complete the task. Agent Drive represents closed-loop sequential decision making with environment feedback; attacks induce unsafe control behaviors, e.g., a sudden stop, through small perturbations that compound over time. Across all tasks, BackdoorAgent provides standardized task loaders, agent templates, and logging and evaluation scripts, while remaining extensible.

## 5 Experiments

We conduct a systematic empirical study with BackdoorAgent to characterize how backdoor vulnerabilities manifest in multi-step agent workflows.

### 5.1 Experimental Setup

**Agent Systems.** We evaluate BackdoorAgent on four representative agent workflows: Agent QA, Agent Code, Agent Drive, and Agent Web. These workflows span complementary regions of the planning–memory–tools design space and differ in both interaction structure and task objectives. Because Agent Web requires multimodal perception, it is evaluated only on backbones that support multimodal inputs, and its results are reported separately in Table 3.

**Backdoor Attacks.** Across all tasks, we evaluate seven representative backdoor attacks (Table 1) spanning three injection channels: planning (BadChain, BadAgent), memory (PoisonedRAG, TrojanRAG, AgentPoison), and tools/environment (AdvAgent, DemonAgent). All attacks are implemented as component-local perturbations within the same BackdoorAgent runtime. We evaluate both closed-source and open-source LLM backbones under identical task instances and step budgets (Hurst et al., 2024; Team et al., 2023; Bai et al., 2023; Team et al., 2025; Guo et al., 2025a; Bai et al., 2022; Liu et al., 2024; Leon, 2025).

**Evaluation Metrics.** We evaluate agent robustness under both clean and backdoor settings using three metrics: Firstly, **Clean ACC** measures the task success rate of a benign agent without trigger injection, where each task instance is judged by a task-specific verifier (e.g., exact match for QA, unit tests for Code, task completion for Web, and safety constraints for Drive). To assess backdoor effec-

Table 2: Unified evaluation on closed-source LLM backbones.

Task	Backbone	Clean ACC	BadChain		PoisonedRAG		TrojanRAG		AgentPoison		AdvAgent		DemonAgent	
			ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC
Code	claude_sonnet4_5	75.80	1.04	<b>80.13</b>	80.05	<b>77.86</b>	80.31	79.13	86.52	<b>85.40</b>	27.28	74.15	4.76	<b>81.93</b>
	gemini-3-flash	56.54	41.67	47.71	<b>92.31</b>	47.83	91.43	31.90	76.49	50.36	85.37	52.32	7.05	46.05
	gpt-4o-mini-0718-global	58.95	<b>61.12</b>	50.66	92.30	57.45	<b>92.86</b>	45.30	<b>98.01</b>	56.74	86.71	56.25	3.70	51.28
	gpt-5-mini-0807-global	<b>81.83</b>	51.37	73.21	58.69	75.17	90.71	<b>86.02</b>	78.39	75.18	78.95	<b>74.50</b>	4.29	68.49
	qwen3-max	71.96	58.34	73.20	64.53	72.99	65.71	74.58	84.51	74.45	<b>89.35</b>	71.62	<b>12.50</b>	68.21
QA	claude_sonnet4_5	86.27	16.53	72.13	67.38	77.05	20.17	73.77	28.48	<b>85.54</b>	25.54	76.50	13.41	<b>80.33</b>
	gemini-3-flash-preview	<b>87.50</b>	<b>26.09</b>	<b>86.34</b>	81.61	<b>87.10</b>	<b>40.19</b>	<b>84.55</b>	37.87	67.57	86.05	<b>84.24</b>	15.32	79.35
	gpt-4o-mini-0718-global	71.95	23.91	56.28	91.30	56.28	39.13	45.36	48.87	59.15	<b>95.76</b>	56.53	<b>26.52</b>	56.83
	gpt-5-mini-0807-global	77.64	23.92	59.18	<b>93.48</b>	57.38	31.30	55.74	<b>57.26</b>	44.53	84.78	54.64	17.83	55.19
	qwen3-max	81.87	23.91	54.15	85.96	84.80	25.22	81.10	45.24	78.47	39.13	73.17	8.75	79.23
Drive	claude_sonnet4_5	51.50	13.72	49.05	37.81	49.06	53.65	55.35	39.02	49.06	17.07	52.20	50.37	45.56
	gemini-3-flash-preview	<b>63.75</b>	48.54	<b>64.15</b>	<b>80.01</b>	<b>64.38</b>	<b>95.76</b>	<b>66.03</b>	48.78	<b>60.38</b>	95.12	<b>65.41</b>	10.53	50.77
	gpt-4o-mini-0718-global	39.50	57.66	41.39	77.21	40.67	95.33	43.40	92.68	30.20	87.52	38.12	61.92	32.91
	gpt-5-mini-0807-global	57.51	43.52	52.77	75.61	52.14	95.12	52.20	<b>95.24</b>	47.98	92.51	40.50	82.93	<b>50.88</b>
	qwen3-max	44.53	<b>58.29</b>	37.57	72.43	43.40	92.68	32.71	82.93	28.39	<b>97.38</b>	36.42	<b>85.31</b>	42.03

Table 3: Unified evaluation on Agent Web (multimodal backbones). Agent Web requires multimodal capabilities and is therefore evaluated only on backbones that support multimodal inputs. We report clean-task accuracy (Clean ACC) and, for each attack, attack success rate (ASR) and accuracy under attack (ACC).

Task	Backbone	Clean ACC	BadChain		PoisonedRAG		TrojanRAG		AgentPoison		AdvAgent		DemonAgent	
			ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC
Web	claude_sonnet4_5	98.54	97.39	98.35	0	96.49	0	<b>99.36</b>	0	98.35	0	99.65	0	<b>98.35</b>
	gemini-3-flash	96.20	<b>98.74</b>	97.47	<b>95.16</b>	94.94	<b>97.44</b>	93.67	<b>86.67</b>	92.41	5.32	93.67	<b>95.40</b>	96.25
	gpt-4o-mini-0718-global	99.16	0	98.89	4.52	95.37	3.57	98.96	8.91	<b>98.39</b>	2.53	<b>99.66</b>	4.53	95.41
	qwen3-vl-235b	<b>99.25</b>	0	<b>99.88</b>	2.31	<b>99.34</b>	5.78	96.54	3.21	97.65	<b>13.92</b>	95.64	14.35	96.57

tiveness, we inject triggers at a designated stage (planning, memory, or tools). Secondly, **ASR** is defined as the percentage of triggered instances in which the agent exhibits the attacker-specified behavior. Thirdly, **ACC** under attack measures task success on the same triggered executions using the identical verifier. Results are reported in Tables 2, 4 and 5.

## 5.2 Results and Analyses

### High attack success often coexists with largely preserved task accuracy across agent workflows.

Across all agent workflows and backbones, Tables 2–3 reveal a recurring pattern in which high attack success rates (ASR) are accompanied by only limited degradation in task accuracy (ACC). This behavior is not unique to agent-based systems: similar phenomena have been observed in earlier backdoor studies on single-turn models, where malicious control can be achieved without substantially impairing nominal task performance. Our results indicate that agent workflows largely preserve this characteristic, even in multi-step and interactive settings. More strikingly, in several configurations we observe cases where task accuracy under attack matches or even exceeds the clean baseline (e.g.,

Agent Code with qwen2.5-72b under AgentPoison: ASR 88.34 with ACC 75.89 vs. clean ACC 78.45; Agent QA with kimi-k2 under AgentPoison: ASR 79.83 with ACC 73.77 vs. clean ACC 71.25). This highlights a critical evaluation challenge for agent systems: performance-based metrics alone may fail to reflect the presence or severity of behavioral compromise. When attack success is decoupled from task-level accuracy, agents can appear to function normally—or even improve on benchmark metrics—while executing harmful objectives. Such cases underscore the risk of relying on standard accuracy-centric evaluation when assessing the safety of multi-step agents and motivate the need for behavior- and trajectory-level analyses beyond task completion scores.

### Vulnerability patterns are structured more by injection channel than by task category.

A second key finding is that vulnerability patterns are primarily organized by injection channel rather than by any single task. Memory-channel attacks are consistently effective whenever retrieved content is persistently reintroduced into context. Across Agent QA and Agent Code, PoisonedRAG and TrojanRAG frequently achieve very high ASR on both

Table 4: Unified evaluation on open-source LLM backbones.

Task	Backbone	Clean ACC	BadChain		BadAgent		PoisonedRAG		TrojanRAG		AgentPoison		AdvAgent		DemonAgent	
			ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC
Code	deepseek-r1-671b	25.43	13.89	27.88	8.69	19.74	14.69	16.79	8.57	13.56	19.35	26.17	7.89	30.05	15.31	28.28
	deepseek-v3.2-exp	60.43	19.45	58.39	7.49	57.42	14.53	56.74	31.43	64.71	16.35	58.16	<b>86.48</b>	59.60	12.62	70.45
	kimi-k2	47.59	<b>22.22</b>	52.15	10.35	50.66	36.17	49.64	<b>84.29</b>	52.59	68.00	50.34	40.91	46.31	12.83	51.20
	qwen2.5-72b-instruct	78.45	18.31	<b>72.67</b>	9.64	74.55	56.36	71.28	62.86	75.21	<b>88.34</b>	<b>75.89</b>	51.35	79.33	<b>17.37</b>	81.25
	qwen3-235b-a22b	<b>80.47</b>	16.54	63.24	<b>10.45</b>	<b>76.31</b>	<b>58.42</b>	<b>81.53</b>	58.57	<b>76.32</b>	85.64	75.41	49.40	<b>82.73</b>	15.45	<b>83.36</b>
QA	deepseek-r1-671b	58.35	12.17	48.63	10.37	38.70	25.22	37.16	16.52	38.80	55.81	58.65	14.35	48.63	19.38	46.70
	deepseek-v3.2-exp	59.50	14.35	57.92	12.62	58.41	<b>56.52</b>	59.02	56.37	45.90	50.38	58.72	34.78	56.83	10.56	56.95
	kimi-k2	71.25	<b>18.70</b>	<b>75.96</b>	<b>23.92</b>	<b>77.43</b>	54.35	<b>71.38</b>	<b>73.91</b>	55.74	<b>79.83</b>	56.39	54.35	<b>73.77</b>	14.36	<b>77.61</b>
	qwen2.5-72b-instruct	69.47	12.17	68.86	15.64	69.22	21.74	66.67	30.43	<b>58.47</b>	36.43	<b>68.26</b>	<b>93.89</b>	72.13	<b>25.37</b>	66.91
	qwen3-235b-a22b	<b>71.46</b>	16.52	67.38	13.39	66.19	15.72	57.54	23.04	57.64	32.84	64.32	91.30	62.46	19.17	64.62
Drive	deepseek-r1-671b	40.50	92.58	41.51	20.36	48.62	<b>77.50</b>	46.25	31.71	41.45	39.52	45.84	80.48	47.11	61.90	<b>53.74</b>
	deepseek-v3.2-exp	48.35	<b>97.85</b>	37.74	24.30	49.95	22.50	45.63	85.37	42.77	78.05	42.99	<b>97.56</b>	38.36	65.03	47.59
	kimi-k2	<b>54.02</b>	92.15	45.80	23.41	47.87	54.53	48.32	49.29	<b>56.63</b>	57.48	<b>53.23</b>	85.71	48.96	<b>75.37</b>	44.45
	qwen2.5-72b-instruct	45.30	97.33	46.25	<b>27.47</b>	48.24	14.87	46.48	<b>87.80</b>	52.77	<b>80.49</b>	47.74	87.21	49.95	61.90	37.32
	qwen3-235b-a22b	50.35	96.47	<b>50.22</b>	22.41	<b>52.04</b>	22.19	<b>52.83</b>	70.77	54.93	31.46	52.87	75.34	<b>51.97</b>	59.99	52.25

Table 5: Unified evaluation on More Advanced LLM backbones.

Task	Backbone	Clean ACC	BadChain		PoisonedRAG		TrojanRAG		AgentPoison		AdvAgent		DemonAgent	
			ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC
Code	gpt-5.2-1211-global	84.73	47.68	79.14	54.82	79.37	85.94	88.27	74.91	78.42	76.58	77.03	3.84	72.91
	gpt-4o-1120-global	62.47	58.63	61.28	90.17	57.64	92.08	45.83	95.61	54.92	83.79	58.11	3.27	54.38
	gemini-3-pro-preview	73.06	34.91	71.22	70.63	74.19	78.12	75.37	68.44	72.11	77.48	69.83	6.37	70.24
QA	gpt-5.2-1211-global	89.37	20.41	64.23	90.78	58.66	27.83	60.14	55.62	47.96	82.43	58.12	15.37	61.84
	gpt-4o-1120-global	74.83	21.36	59.74	92.63	57.31	29.02	55.18	56.49	45.07	94.52	54.31	17.14	58.07
	gemini-3-pro-preview	90.61	18.92	70.84	84.19	68.57	22.46	74.52	43.81	67.14	78.76	71.02	12.61	73.24
Drive	gpt-5.2-1211-global	60.73	40.92	55.37	72.61	55.14	88.77	52.86	90.43	52.19	88.71	54.23	80.48	56.37
	gpt-4o-1120-global	44.91	53.86	42.31	77.28	41.53	92.54	43.61	91.83	34.47	90.52	39.11	84.97	34.82
	gemini-3-pro-preview	70.46	42.61	69.14	79.43	65.02	91.57	66.21	54.63	63.08	88.94	67.01	10.72	62.83

open and closed source backbones (often  $> 90$ , e.g., Agent QA with qwen2.5-72b under AdvAgent: ASR 93.89; Agent Code with gpt-4o-mini under AgentPoison: ASR 98.01), while maintaining usable task accuracy. These attacks directly support the attacker’s objectives—semantic misinformation in QA and destructive operations in Code—by repeatedly reinforcing poisoned content across steps. Planning-channel attacks show moderate but stable effectiveness in iterative workflows. In contrast, tool and environment channel attacks dominate in closed workflow settings: in Agent Drive, manipulating tool feedback or environment observations is sufficient to induce unsafe behaviors with very high ASR (commonly  $> 90$  for BadChain and  $> 80$  for AdvAgent across open-source backbones), even without corrupting internal reasoning or memory.

**Model performance and backdoor robustness diverge in agent-based systems.** Stronger model backbones do not consistently translate into greater resistance to agent backdoors. Across multiple workflows, models that achieve high clean-task accuracy remain highly susceptible once back-

door triggers are introduced. For example, high-performing backbones in Agent QA and Agent Code continue to exhibit high attack success rates under memory-based attacks, while in Agent Drive, open-source models can be nearly fully compromised by relatively simple planning- or tool-based perturbations. These failures occur despite substantial differences in model scale and clean performance. Taken together, these observations suggest that agent-level vulnerability is driven less by the expressive power of the underlying model and more by workflow-level properties, such as the persistence of intermediate artifacts, the reuse of feedback across steps, and the amplification effects of closed-loop interaction. Consequently, improving backbone performance alone is insufficient as a defense strategy, and robust agent design must account for how information flows and accumulates throughout the agent’s execution.

**In sequential agents, small backdoor perturbations can propagate into large behavioral deviations.** Comparing Agent Drive with Agent QA and Agent Code shows a distinct amplification ef-

Table 6: **Module-wise ASR averaged across all attacks and tasks.** Values are computed by aggregating results from Tables 2 and 4.

Backbone	Planning ASR	Memory ASR	Tools ASR
gpt-family	43.58	77.97	60.28
claude-family	10.43	54.82	23.07
gemini-family	39.49	75.39	48.98
qwen-family	35.41	55.45	54.45
deepseek	27.84	38.91	42.20

Table 7: **Memory attack success rate (ASR) under different memory poisoning ratios.**

Backbone	Memory ASR (0.1)	Memory ASR (0.2)	Memory ASR (0.5)
gpt-family	53.06	77.97	97.72
gemini-family	50.39	75.39	97.00
qwen-family	33.25	55.45	86.75
deepseek	21.84	38.91	70.83

fect in sequential decision-making. In QA and Code, attacks mainly steer intermediate text artifacts, and their impact is often bounded within a step; planning attacks are typically moderate. In contrast, Agent Drive is vulnerable to cascading failures because each perturbed plan or tool feedback changes the next state, compounding over time. Empirically, BadChain reaches ASR above 90 on every open-source backbone, and AdvAgent frequently exceeds 90 on both closed- and open-source Drive settings, a pattern not matched as consistently in QA/Code. This indicates that sequential state transitions turn small perturbations into long-horizon derailments.

Overall, our results show that agent backdoors enable reliable, objective-level control in multi-step workflows, inducing task-specific harms while preserving apparent task performance. This decoupling between behavioral correctness and controllability highlights the need for trajectory-level, workflow-aware evaluation beyond standard accuracy metrics.

### 5.3 Ablation Studies

**Family-Level Aggregation of Module Vulnerability.** To assess whether vulnerability trends persist beyond individual tasks and attacks, we aggregate results at the model-family level. Following Table 1, attacks are grouped by injection channel into planning, memory, and tool-based categories, and we compute the average ASR for each category across all tasks. As shown in Table 6, memory-based attacks consistently achieve the highest ASR across nearly all model families, indicating that retrieval and persistent memory constitute a system-

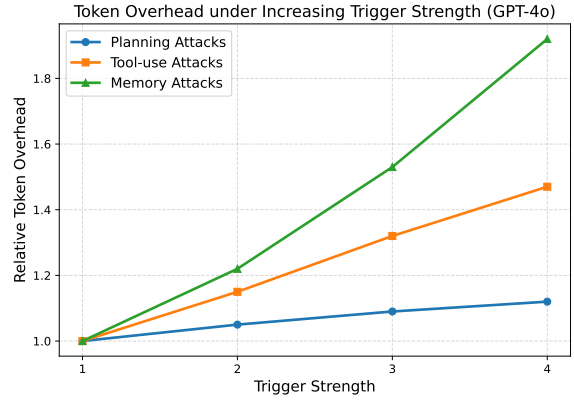


Figure 3: **Token overhead vs. trigger strength.**

atic attack surface. Planning-based attacks exhibit lower but relatively stable ASR, while tool-based attacks show greater variability across families, reflecting their dependence on task dynamics and environment interaction. Besides, we have added an ablation that varies the poisoning ratio in the external memory / retrieval corpus and measures the resulting Memory-channel ASR, aggregated at the model-family level. As shown in the Table 7, these results show a monotonic increase in attack success as poisoning ratio increases, suggesting that memory-channel vulnerabilities are sensitive to corpus-level contamination intensity.

**Efficiency Across Attack Modules.** We further analyze the computational cost of different attack modules by measuring total token consumption over full agent trajectories. Figure 3 shows that memory-based attacks incur the highest token overhead as trigger strength increases, due to repeated retrieval and reinsertion of memory content. Planning-based attacks remain the most token-efficient, as they primarily manipulate transient reasoning traces. This highlights a potential trade-off between attack effectiveness and efficiency in practical, resource-constrained deployments.

### 5.4 Exploring Potential Defenses

We conduct a preliminary study to examine whether probability-based backdoor detection signals developed for standalone LLMs transfer to agentic settings. Specifically, we adopt the token-probability analysis proposed in CleanGen (Li et al., 2024c; Wang et al., 2025c; Yi et al., 2025) and apply it to multi-step agent executions, comparing output-token probabilities under clean and triggered conditions. Figure 4 reports the average probabilities assigned to attack-target tokens and non-target

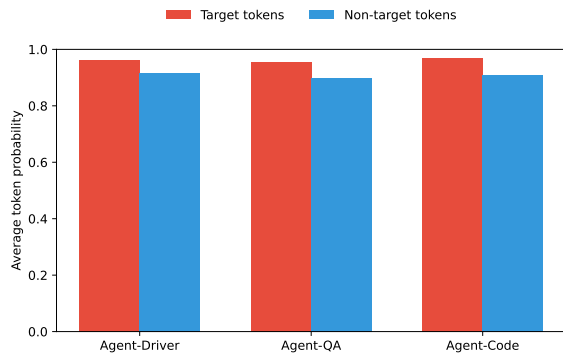


Figure 4: **Average token probabilities for target vs. non-target tokens.**

tokens across different agent tasks. While target tokens are assigned slightly higher probabilities on average, the difference is small and inconsistent across settings, even when backdoors successfully control agent behavior. These results suggest that probability-based cues effective for single-turn LLM backdoor detection do not directly generalize to agent workflows. In multi-step agents, malicious effects can be delayed and interleaved with benign reasoning, memory retrieval, and tool outputs, weakening token-level probability signals. This highlights the need for agent defenses that reason over trajectories rather than isolated model outputs.

## 6 Conclusion

We introduced BackdoorAgent, a framework and benchmark for analyzing backdoor vulnerabilities in multi-step LLM agent workflows. Across four representative agent settings and diverse closed and open source backbones, we find that backdoors are primarily workflow-level phenomena: memory-channel attacks are the most persistent, tool and environment-channel attacks dominate closed-loop agents, and strong clean performance does not imply robustness. Many attacks preserve high task accuracy while reliably inducing harmful behaviors, exposing a stealthy and practically concerning failure mode. We will release our code and dataset to support future research on backdoor attacks and their defenses in agentic systems.

## Limitations

BackdoorAgent provides a modular and stage-aware framework for analyzing backdoor threats in LLM agents; however, several limitations remain. While our experiments cover multiple representative agent tasks, attack types, and model backbones,

they do not fully capture the diversity of real-world agent architectures, execution environments, and threat models. Moreover, our analysis primarily examines trigger persistence and cross-stage propagation under controlled benchmark settings. Additional work is needed to study adaptive adversaries and more complex deployment scenarios, where backdoor behaviors may manifest in richer and less predictable ways. We will continuously update our benchmark and open-source all code to support reproducibility and future research.

## Acknowledgments

This work was supported by the National Natural Science Foundation of China (General Program, Grant No. 72571281), the Excellent Young Scientist Fund of Hunan Province (Grant No. 2025JJ40066).

## References

- EU Artificial Intelligence Act. 2024. The eu artificial intelligence act. *European Union*.
- Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, and 1 others. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.
- Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, and 1 others. 2022. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*.
- Hyungjoo Chae, Namyoun Kim, Kai Tzu-iunn Ong, Minju Gwak, Gwanwoo Song, Jihoon Kim, Sunghwan Kim, Dongha Lee, and Jinyoung Yeo. 2024. Web agents with world models: Learning and leveraging environment dynamics in web navigation. *arXiv preprint arXiv:2410.13232*.
- Li Changjiang, Liang Jiacheng, Cao Bochuan, Chen Jinghui, and Wang Ting. 2025. Your agent can defend itself against backdoor attacks. *arXiv preprint arXiv:2506.08336*.
- Zhaorun Chen, Zhen Xiang, Chaowei Xiao, Dawn Song, and Bo Li. 2024. Agentpoison: Red-teaming llm agents via poisoning memory or knowledge bases. *NeurIPS*.
- Pengzhou Cheng, Yidong Ding, Tianjie Ju, Zongru Wu, Wei Du, Ping Yi, Zhuosheng Zhang, and Gongshen Liu. 2024. Trojanrag: Retrieval-augmented generation can be backdoor driver in large language models. *arXiv preprint arXiv:2405.13401*.
- Can Cui, Zichong Yang, Yupeng Zhou, Yunsheng Ma, Juanwu Lu, Lingxi Li, Yaobin Chen, Jitesh Panchal, and Ziran Wang. 2024. Personalized autonomous driving with large language models: Field experiments. In *ITSC*. IEEE.
- Huaizhi Ge, Yiming Li, Qifan Wang, Yongfeng Zhang, and Ruixiang Tang. 2025. When backdoors speak: Understanding llm backdoor attacks through model-generated explanations. In *ACL*.
- Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025a. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.
- Qi Guo, Xiaojun Jia, Shanmin Pang, Simeng Qin, Lin Wang, Ju Jia, Yang Liu, and Qing Guo. 2025b. Physpatch: A physically realizable and transferable adversarial patch attack for multimodal large language models-based autonomous driving systems. *arXiv preprint arXiv:2508.05167*.
- Jiuzhou Han, Wray Buntine, and Ehsan Shareghi. 2024. Towards uncertainty-aware language agent. In *ACL*.
- Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, and 1 others. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.
- Jakub Lála, Odhran O’Donoghue, Aleksandar Shtedritski, Sam Cox, Samuel G Rodrigues, and Andrew D White. 2023. Paperqa: Retrieval-augmented generative agent for scientific research. *arXiv preprint arXiv:2312.07559*.
- Maikel Leon. 2025. Gpt-5 and open-weight large language models: Advances in reasoning, transparency, and control. *Information Systems*, page 102620.
- Guopeng Li, Ruiqi Wu, and Haisheng Tan. 2025a. A plan reuse mechanism for llm-driven agent. *arXiv preprint arXiv:2512.21309*.
- Xinyi Li, Sai Wang, Siqi Zeng, Yu Wu, and Yi Yang. 2024a. A survey on llm-based multi-agent systems: workflow, infrastructure, and challenges. *Vicinatearth*.
- Yige Li, Hanxun Huang, Yunhan Zhao, Xingjun Ma, and Jun Sun. 2024b. Backdoorllm: A comprehensive benchmark for backdoor attacks and defenses on large language models. *arXiv preprint arXiv:2408.12798*.
- Yige Li, Zhe Li, Wei Zhao, Nay Myat Min, Hanxun Huang, Xingjun Ma, and Jun Sun. 2025b. Autobackdoor: Automating backdoor attacks via llm agents. *arXiv preprint arXiv:2511.16709*.
- Yuetai Li, Zhangchen Xu, Fengqing Jiang, Luyao Niu, Dinuka Sahabandu, Bhaskar Ramasubramanian, and Radha Poovendran. 2024c. Cleangen: Mitigating backdoor attacks for generation tasks in large language models. *arXiv preprint arXiv:2406.12257*.
- Aishan Liu, Yuguang Zhou, Xianglong Liu, Tianyuan Zhang, Siyuan Liang, Jiakai Wang, Yanjun Pu, Tianlin Li, Junqi Zhang, Wenbo Zhou, and 1 others. 2025a. Compromising llm driven embodied agents with contextual backdoor attacks. *IEEE Transactions on Information Forensics and Security*.
- Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, and 1 others. 2024. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.
- Chengwei Liu, Lyuye Zhang, Xiufeng Xu, Wenbo Guo, and Yang Liu. 2025b. Towards the versioning of llm-agent-based software. In *Proc. ESEC/FSE*.
- Xingjun Ma, Yifeng Gao, Yixu Wang, Ruofan Wang, Xin Wang, Ye Sun, Yifan Ding, Hengyuan Xu, Yunhao Chen, Yunhan Zhao, and 1 others. 2025. Safety at scale: A comprehensive survey of large model safety. *arXiv preprint arXiv:2502.05206*.

- Jiageng Mao, Junjie Ye, Yuxi Qian, Marco Pavone, and Yue Wang. 2023. A language agent for autonomous driving. *arXiv preprint arXiv:2311.10813*.
- Tula Masterman, Sandi Besen, Mason Sawtell, and Alex Chao. 2024. The landscape of emerging ai agent architectures for reasoning, planning, and tool calling: A survey. *arXiv preprint arXiv:2404.11584*.
- Wenqi Shi, Ran Xu, Yuchen Zhuang, Yue Yu, Jieyu Zhang, Hang Wu, Yuanda Zhu, Joyce C. Ho, Carl Yang, and May Dongmei Wang. 2024. EHRAgent: Code empowers large language models for few-shot complex tabular reasoning on electronic health records. In *EMNLP*. Association for Computational Linguistics.
- Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, and 1 others. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.
- Kimi Team, Yifan Bai, Yiping Bao, Guanduo Chen, Jiahao Chen, Ningxin Chen, Ruijue Chen, Yanru Chen, Yuankun Chen, Yutian Chen, and 1 others. 2025. Kimi k2: Open agentic intelligence. *arXiv preprint arXiv:2507.20534*.
- Khanh-Tung Tran, Dung Dao, Minh-Duong Nguyen, Quoc-Viet Pham, Barry O’Sullivan, and Hoang D Nguyen. 2025. Multi-agent collaboration mechanisms: A survey of llms. *arXiv preprint arXiv:2501.06322*.
- Haowei Wang, Junjie Wang, Xiaojun Jia, Rupeng Zhang, Mingyang Li, Zhe Liu, Yang Liu, and Qing Wang. 2025a. Adinject: Real-world black-box attacks on web agents via advertising delivery. *arXiv preprint arXiv:2505.21499*.
- Kaixiang Wang, Zhaojiacheng Zhou, Bunyod Suvonov, Jiong Lou, and Jie Li. 2025b. Agentshield: Make mas more secure and efficient. *arXiv preprint arXiv:2511.22924*.
- Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, and 1 others. 2024a. A survey on large language model based autonomous agents. *Frontiers of Computer Science*.
- Yifei Wang, Dizhan Xue, Shengjie Zhang, and Shengsheng Qian. 2024b. Badagent: Inserting and activating backdoor attacks in llm agents. *arXiv preprint arXiv:2406.03007*.
- Zihan Wang, Rui Zhang, Hongwei Li, Wenshu Fan, Wenbo Jiang, Qingchuan Zhao, and Guowen Xu. 2025c. Configuard: A simple and effective backdoor detection for large language models. *arXiv preprint arXiv:2508.01365*.
- Yuan-An Xiao, Pengfei Gao, Chao Peng, and Yingfei Xiong. 2025. Improving the efficiency of llm agent systems through trajectory reduction. *arXiv preprint arXiv:2509.23586*.
- Chejian Xu, Mintong Kang, Jiawei Zhang, Zeyi Liao, Lingbo Mo, Mengqi Yuan, Huan Sun, and Bo Li. 2024. Advagent: Controllable blackbox red-teaming on web agents. *arXiv preprint arXiv:2410.17401*.
- John Yang, Carlos E Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. 2024a. Swe-agent: Agent-computer interfaces enable automated software engineering. *NeurIPS*.
- 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. *NeurIPS*.
- Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. In *ICLR*.
- Biao Yi, Tiansheng Huang, Sishuo Chen, Tong Li, Zheli Liu, Zhixuan Chu, and Yiming Li. 2025. Probe before you talk: Towards black-box defense against backdoor unalignment for large language models. *arXiv preprint arXiv:2506.16447*.
- Yangyang Yu, Haohang Li, Zhi Chen, Yuechen Jiang, Yang Li, Jordan W Suchow, Denghui Zhang, and Khaldoun Khashanah. 2025. Finmem: A performance-enhanced llm trading agent with layered memory and character design. *IEEE Transactions on Big Data*.
- Jianhao Yuan, Shuyang Sun, Daniel Omeiza, Bo Zhao, Paul Newman, Lars Kunze, and Matthew Gadd. 2024. Rag-driver: Generalisable driving explanations with retrieval-augmented in-context learning in multi-modal large language model. *arXiv preprint arXiv:2402.10828*.
- Guancheng Zeng, Xueyi Chen, Jiawang Hu, Shao-hua Qi, Yaxuan Mao, Zhantao Wang, Yifan Nie, Shuang Li, Qiuyang Feng, Pengxu Qiu, and 1 others. 2025. Routine: A structural planning framework for llm agent system in enterprise. *arXiv preprint arXiv:2507.14447*.
- Xiaoyu Zhang, Yi Wang, Along He, Haobin Wang, and Tao Li. 2025a. Tape: A multi-agent framework for task-adaptive planning and execution in resource-constrained environments. *Expert Systems with Applications*.
- Xiaoyu Zhang, Cen Zhang, Tianlin Li, Yihao Huang, Xiaojun Jia, Ming Hu, Jie Zhang, Yang Liu, Shiqing Ma, and Chao Shen. 2025b. Jailguard: A universal detection framework for prompt-based attacks on llm systems. *ACM Transactions on Software Engineering and Methodology*.

Yiran Zhang, Ruiyin Li, Peng Liang, Weisong Sun, and Yang Liu. 2025c. Knowledge-based multi-agent framework for automated software architecture design. In *Proc. ESEC/FSE*, pages 530–534.

Jianguo Zhao, Yuqiang Sun, Cheng Huang, Chengwei Liu, YaoHui Guan, Yutong Zeng, and Yang Liu. 2025. Towards secure code generation with llms: A study on common weakness enumeration. *IEEE Transactions on Software Engineering*.

Pengyu Zhu, Zhenhong Zhou, Yuanhe Zhang, Shilinlu Yan, Kun Wang, and Sen Su. 2025. Demona-gent: Dynamically encrypted multi-backdoor implantation attack on llm-based agent. *arXiv preprint arXiv:2502.12575*.

Wei Zou, Runpeng Geng, Binghui Wang, and Jinyuan Jia. 2025. {PoisonedRAG}: Knowledge corruption attacks to {Retrieval-Augmented} generation of large language models. In *USENIX Security*.

## A Additional Defenses

While BackdoorAgent is primarily designed as an attack-centric benchmark, it also provides an opportunity to examine whether existing backdoor defense signals—largely developed for standalone LLMs—remain meaningful in agentic settings. In this section, we do not propose new defense mechanisms. Instead, we analyze the transferability and limitations of representative LLM backdoor defense ideas when applied to multi-step, tool-augmented agents.

Most existing backdoor defenses (Li et al., 2024c; Wang et al., 2025c; Yi et al., 2025) for LLMs rely on identifying abnormal patterns in single-pass generation, such as sharp probability spikes on trigger tokens, distributional shifts in logits, or inconsistencies revealed by probing prompts. These signals implicitly assume a static input–output mapping, where malicious behavior manifests directly in the model’s immediate response. In contrast, LLM-based agents operate through recurrent loops involving planning, memory retrieval, and tool interaction. Malicious effects may only emerge after multiple steps, and intermediate signals are repeatedly mixed with benign context, retrieved documents, and external observations. As a result, the statistical footprints exploited by prior LLM backdoor defenses may be diluted or obscured.

To assess this gap, we conduct a preliminary analysis inspired by probability-based LLM defenses, examining whether token-level probability differences can reliably separate clean and backdoored agent executions. Specifically, we compare the output-token probability distributions of a target agent backbone against a reference model under both clean and triggered conditions.

Figure 5 shows the resulting ROC curve aggregated across agent tasks and attack types. Although the detector achieves an AUROC above random guessing, the margin is modest, indicating limited separability.

Our results suggest that defenses effective for standalone LLM backdoors do not directly generalize to agentic systems. Even when backdoors successfully control long-horizon agent behavior, their probabilistic signatures may be subtle, delayed, or entangled with benign reasoning, retrieval, and tool outputs. This fundamental mismatch explains why simple probability-based detectors struggle to achieve high confidence in the agent setting. These findings reinforce the central motivation of Back-

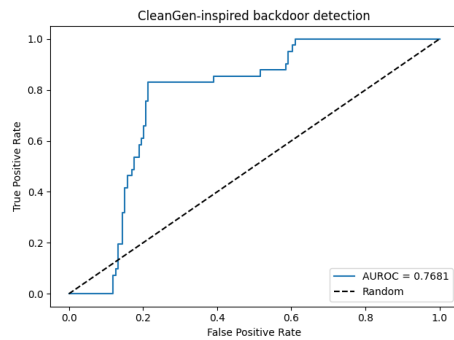


Figure 5: **ROC curve of probability-based detection in agent outputs.** While backdoored trajectories exhibit some detectable signal, the separation remains weak, highlighting the difficulty of directly transferring LLM defenses to agent settings.

doorAgent. Agent backdoors should not be treated as a straightforward extension of LLM backdoors. Instead, they introduce new challenges related to temporal propagation, cross-module interactions, and environment feedback. We hope this benchmark will facilitate the development of agent-aware defense strategies that go beyond single-step output analysis and explicitly reason about multi-step trajectories, state evolution, and tool-mediated effects.