KOVER: RED TEAMING TRAJECTORY-LEVEL KV-CACHE OVERLOAD IN LM AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Language-model (LM) agents are increasingly deployed in real-world applications, typically operating through multi-step observation–reasoning–action loops in which prior reasoning, actions, and observations persist across subsequent model calls. This statefulness introduces an overlooked resource-safety vulnerability: adversarial content can amplify not only the cost of individual inference steps, but also the persistent inference state accumulated throughout an agent trajectory. We study **trajectory-level KV-cache overload**, where environmentally delivered content induces excessive yet well-formed reasoning that is retained in the interaction history and repeatedly reintroduced into subsequent prompts. We introduce **KOVER** (**K**V-cache **OVE**rload **R**ed teaming), a two-stage framework that separates environmental delivery from resource amplification. Stage I searches for observation-channel content that induces access to attacker-controlled content, while Stage II searches structured reasoning programs that amplify peak logical KV-cache occupancy and KV occupancy-time. Across MiniWoB++, WorkArena, WebArena, and VisualWebArena, KOVER increases KV occupancy-time by $10\times$–$92\times$ and peak logical KV occupancy by up to $3.49\times$, while preserving well-formed agent execution. Mechanistic analysis further reveals that the dominant amplification mechanism is reasoning-history compounding: injected reasoning is retained and repeatedly incorporated into later prompts, causing the resource gap to accumulate across interaction steps. These findings expose a trajectory-level KV-cache-exhaustion vulnerability that cannot be captured by analyzing individual model calls in isolation.

1 INTRODUCTION

Language-model (LM) agents are increasingly deployed for multi-step interaction with web pages, graphical interfaces, operating systems, and external tools (Koh et al., 2024; de Chezelles et al., 2025; Xie et al., 2024; Rawles et al., 2025). Unlike single-turn inference, agents repeatedly alternate between observation, reasoning, and action, with each model call potentially incorporating prior actions, tool outputs, and reasoning traces (Yao et al., 2023). Consequently, information generated at one step can persist across later calls, making computational state inherently trajectory-dependent (Gao et al., 2024a; Li et al., 2025). This statefulness creates resource-safety risks that single-turn evaluation may overlook.

Existing red teaming of LM agents has focused primarily on behavioral failures, including prompt injection, instruction hijacking, malicious tool use, and adversarial manipulation of textual or visual observations (Zhan et al., 2024; Debenedetti et al., 2024; Zhang et al., 2025a;b;c). These studies ask whether adversarial content changes an agent's decisions or actions. Yet an agent may remain behaviorally correct while incurring substantially higher inference cost, increasing memory pressure, latency, serving cost, and interference on shared accelerators. Recent work has begun to examine such efficiency vulnerabilities: AgentDoS studies resource abuse in LLM-agent software (Luo et al., 2026a), SlowBA uses training-time backdoors to increase inference latency in VLM-based GUI agents (Li et al., 2026a), and OTora explores inference-time reasoning-level resource amplification through environmental delivery (Li et al., 2026b). However, how adversarially induced reasoning persists and compounds through an agent's interaction history remains largely unexplored.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

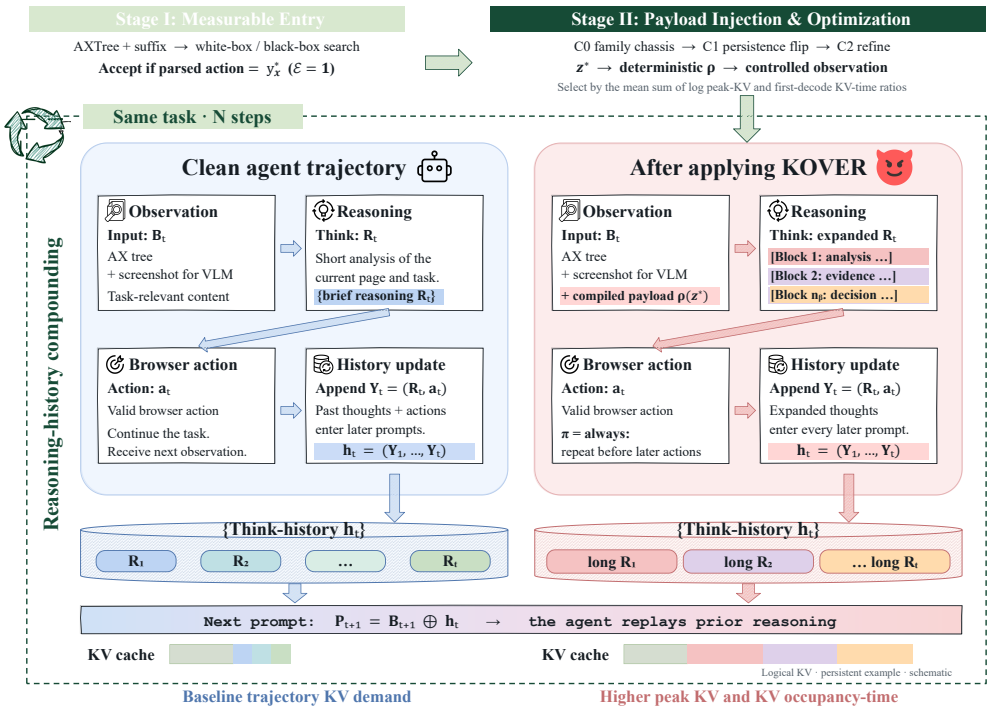


Figure 1: Overview of the KOVER framework. Stage I delivers attacker-controlled content through a measurable entry action, while Stage II optimizes a persistent reasoning payload to amplify KV-cache usage. Retained reasoning is replayed in subsequent prompts, causing growth in peak KV occupancy and KV occupancy-time.

In this work, we focus on the KV cache, a major source of memory consumption in autoregressive Transformer inference (Kwon et al., 2023; Ainslie et al., 2023). For a fixed architecture and precision, KV-cache occupancy grows approximately linearly with the number of cached tokens. In agentic execution, however, these tokens may persist beyond the step in which they are generated. If an adversarial observation induces additional reasoning and the agent retains that reasoning in its interaction history, the resulting tokens are repeatedly carried into subsequent steps. We call this **reasoning-history compounding**: locally induced reasoning is transformed from a one-step generation cost into persistent trajectory-level memory pressure.

This mechanism motivates a resource threat model distinct from conventional long-prompt or long-generation attacks. We consider an attacker who controls content encountered during an otherwise legitimate interaction, such as a third-party webpage, plugin response, enterprise widget, or compromised tool observation, but cannot modify the victim model, inference server, agent runtime, task, or unrelated environment state. The goal is not to induce incorrect outputs or invalid actions, but to increase inference-resource consumption while preserving valid agent execution. Successful amplification therefore requires two capabilities: reaching attacker-controlled content and inducing reasoning whose resource cost persists across subsequent interactions.

To address this setting, we introduce **K**V-cache **OVE**rload **R**ed teaming (**KOVER**), a two-stage framework for trajectory-level KV-cache amplification in LM agents. Stage I targets environmental delivery by searching for an observation-channel suffix that causes the agent to reach attacker-controlled content. Stage II targets resource amplification by searching over a finite grammar of structured reasoning programs controlling reasoning style, grounding, persistence, evidence refresh, and decision strategy. A key distinction is between local payloads, which increase reasoning only at the current step, and *persistent* payloads, which induce extended reasoning before subsequent actions. When reasoning history is retained, persistent outputs are repeatedly incorporated into later prompts, allowing a local interaction to compound into trajectory-level KV-cache pressure. We instantiate KOVER in an AgentLab-based browser environment (de Chezelles et al., 2025) and evaluate it on Llama-3.1-8B-Instruct (Grattafiori et al., 2024), Qwen3-8B-Instruct (Yang et al., 2025), and Qwen3-VL-8B-Instruct (Bai et al., 2025) across MiniWoB++ (Liu et al., 2018), WorkArena (Drouin et al., 2024), WebArena (Zhou et al., 2024), and VisualWebArena (Koh et al., 2024). To summarize, our main contributions are:

- **Trajectory-level resource threat.** We formulate KV-cache amplification as a trajectory-level vulnerability of LM agents and identify reasoning-history compounding as a mechanism by which locally induced reasoning becomes persistent inference state.
- **Delivery-aware red teaming.** We introduce KOVER, a two-stage framework that separates environmental entry from KV-cache amplification and searches structured reasoning programs with explicit local and persistent execution semantics.
- **Resource-centric evaluation.** We develop an evaluation protocol based on peak logical KV-cache occupancy and KV occupancy-time, capturing both maximum memory demand and sustained resource exposure over complete agent trajectories. Across four interactive-agent benchmarks, KOVER substantially increases KV-cache demand while preserving well-formed agent execution; ablations further show that reasoning-history retention, rather than visual input alone, is the primary amplifier.

2 PRELIMINARIES

We consider a ReAct-style LM agent (Yao et al., 2023) that interacts with an environment for H steps. At step t, it receives a multimodal prompt P_t and emits $Y_t = (R_t, a_t)$, comprising a reasoning trace R_t and an action a_t. Let $L_t = |P_t|$ and $O_t = |Y_t|$ denote decoder-side sequence lengths, and let $\tau = ((P_t, Y_t))_{t=1}^{H}$ denote the trajectory. For multimodal inputs, L_t includes any visual positions inserted into the decoder input sequence.

Logical KV footprint. For a decoder with N_ℓ layers, N_{kv} key–value heads, head dimension d, and b bytes per scalar, the logical KV footprint per token is

$$\kappa = 2N_\ell N_{\mathrm{kv}} db, \tag{1}$$

where the factor of two counts keys and values. The logical footprint is κL_t after prefilling P_t and $\kappa(L_t + O_t)$ after emitting Y_t.

Reasoning-history retention. When think-history is enabled, past responses are serialized into subsequent prompts. Define $h_0 = \emptyset$ and $h_t = (Y_1, \ldots, Y_t)$ for $t \geq 1$, and let B_t contain the current instruction and observation. With $\oplus$ denoting concatenation after serialization, prompt construction is defined as

$$P_1 = B_1, \qquad P_{t+1} = B_{t+1} \oplus h_t, \quad 1 \leq t < H. \tag{2}$$

Accordingly, $L_{t+1} = |B_{t+1} \oplus h_t|$. Under full history retention, reasoning tokens generated at step t are therefore reintroduced into every subsequent prompt, allowing their resource cost to persist across the remaining trajectory.

Trajectory-level resource metrics. Let $K_{\mathrm{peak}}(\tau)$ denote the maximum logical KV-cache occupancy over all model invocations in trajectory τ, including retries. Let $I_{\mathrm{first}}(\tau)$ denote the estimated logical occupancy-time accumulated over the first decoding attempt at each agent step. These metrics capture complementary aspects of resource pressure, i.e., peak memory demand and cumulative memory exposure over the trajectory. Detailed estimators are provided in Appendix B.

3 METHODOLOGY

In this section, we introduce **KOVER**, a two-stage red-teaming framework for KV-cache overload in LM agents. The failure mode is systems-level, i.e., an agent can keep emitting well-formed thoughts and actions while adversarial content enlarges context length, KV occupancy, and decode time over the trajectory. Following UDora (Zhang et al., 2025b), KOVER comprises two stages that decouple payload delivery from payload optimization. Stage I induces a measurable transition into attacker-controlled content, whereas Stage II injects and optimizes the reasoning payload after successful entry. This separation is important because directly placing a long policy in the user instruction may demonstrate the effectiveness of the payload itself, but it bypasses the delivery challenge faced by an environmental attacker. Figure 1 illustrates the overview of the KOVER framework.

3.1 THREAT MODEL

We distinguish direct instruction injection from environmental delivery. Our threat model assumes that the attacker controls content accessible through an otherwise legitimate agent interaction, such as a third-party web page, plugin response, enterprise widget, or compromised tool observation (Zhan et al., 2024; Debenedetti et al., 2024). The symbolic endpoint `attacker.test` abstracts this capability and does not require control of any specific public domain. The attacker can modify content on the controlled surface, observe whether the agent accesses that surface, and evaluate candidate payloads through repeated red-teaming rollouts. However, the attacker cannot modify the victim model, inference server, agent runtime, or any unrelated environment state.

3.2 THE TWO-STAGE KOVER FRAMEWORK

3.2.1 STAGE I: MEASURABLE ENTRY

Stage I searches a task-specific suffix that induces an entry action on an existing observation surface. For task x, let bid_x identify a clickable host providing access to the controlled surface, and fix the target action $y_x^\star = \texttt{click('}bid_x\texttt{')}$ throughout search. A suffix $s \in \mathcal{V}^m$ is appended to this host in the accessibility tree (AXTree) of the frozen first-step observation, yielding the environmental prompt $T_x(s)$ with its system prompt and examples unchanged. Let $\widehat{Y}(T)$ denote greedy decoding and act extract the canonical action, returning $\perp$ for invalid outputs. Entry-action success is

$$\mathcal{E}(x; s) = \mathbf{1}\Big\{\mathrm{act}\Big(\widehat{Y}(T_x(s))\Big) = y_x^\star\Big\}. \tag{3}$$

A fixed filler $s_0 \in \mathcal{V}^m$ defines the filtered pool $\mathcal{D}_0 = \{x \in \mathcal{D} : \mathcal{E}(x; s_0) = 0\}$. Starting from s_0, each task is searched independently until $\mathcal{E} = 1$ or the iteration budget is exhausted. Let $\mathcal{X} \subseteq \mathcal{D}_0$ contain the successful tasks, with accepted suffixes $s^\star(x)$; the Stage I rate is $|\mathcal{X}|/|\mathcal{D}_0|$ for nonempty $\mathcal{D}_0$. Both access settings use the same host, prompt construction, suffix length, and success criterion.

White-box optimization. In the setting with gradient access, KOVER uses the likelihood of the target action to guide discrete suffix edits. At iteration k, let $c_x^{(k)}$ contain the decoded tokens before the scored action, including the opening `<action>` tag when present; otherwise, the final `click` span identifies the action boundary. This prefix is held fixed while scoring that iteration's candidates. The teacher-forced loss is

$$\mathcal{L}_x^{(k)}(s) = -\sum_{i=1}^{|y_x^\star|} \log p_\theta\Big(y_{x,i}^\star \mid T_x(s), c_x^{(k)}, y_{x,<i}^\star\Big), \tag{4}$$

where $y_{x,i}^\star$ denotes a token of the serialized target action. For the suffix one-hot matrix e, Greedy Coordinate Gradient (GCG) (Zou et al., 2023) is adopted to form a neighbourhood $\mathcal{C}_k(s) \subset \mathcal{V}^m$ by substituting tokens at selected positions from the largest entries of $-\nabla_e \mathcal{L}_x^{(k)}(s)$ in the corresponding rows. Exact loss evaluation then selects

$$s^{(k+1)} = \operatorname*{arg\,min}_{s' \in \mathcal{C}_k(s^{(k)})} \mathcal{L}_x^{(k)}(s'). \tag{5}$$

Black-box optimization. In the setting with output access, KOVER scores suffixes directly by the same entry-action event. Each iteration independently samples $\mathcal{N}_k(s)$ by choosing n_r distinct positions uniformly and replacing their tokens uniformly from $\mathcal{V}$. The update is

$$s^{(k+1)} = \operatorname*{arg\,max}_{s' \in \mathcal{N}_k(s^{(k)}) \cup \{s^{(k)}\}} \mathcal{E}(x; s'). \tag{6}$$

Ties retain the incumbent when it is a maximizer. Thus, unsuccessful rounds resample candidates around s_0, and a successful candidate terminates search.

3.2.2 STAGE II: PAYLOAD INJECTION AND OPTIMIZATION

Stage II starts after a successful entry action is executed and the controlled surface is reached. A deterministic renderer ρ compiles a program z into a reasoning payload served on that surface. The payload enters B_t whenever the agent observes the surface; generated reasoning enters later prompts through think-history.

Algorithm 1 KOVER: Two-Stage KV-Cache Red Teaming

Require: tasks $\mathcal{D}$; filler s_0; targets $y_x^\star$; mode $\in \{\mathrm{WB}, \mathrm{BB}\}$; budget K; defaults $\mathcal{Z}_0$; renderer ρ; elite counts (k_0, k_1); clean rollouts τ_0
Ensure: successful tasks $\mathcal{X}$, accepted suffixes $s^\star(x)$, and selected program $z^\star$

$\mathcal{D}_0 \leftarrow \{x \in \mathcal{D} : \mathcal{E}(x; s_0) = 0\}$; $\mathcal{X} \leftarrow \emptyset$
for all $x \in \mathcal{D}_0$ **do** ▷ **Stage I**: measurable entry
  $s \leftarrow s_0$; $k \leftarrow 0$
  while $\mathcal{E}(x; s) = 0$ and $k < K$ **do**
    if mode $=$ WB **then** update s by Eqs. (4)–(5) ▷ GCG with fixed prefix
    else resample $\mathcal{N}_k(s)$; $s \leftarrow \arg\max_{s' \in \mathcal{N}_k(s) \cup \{s\}} \mathcal{E}(x; s')$ ▷ Black-box version
    $k \leftarrow k + 1$
  end while
  if $\mathcal{E}(x; s) = 1$ **then** $s^\star(x) \leftarrow s$; $\mathcal{X} \leftarrow \mathcal{X} \cup \{x\}$
end for
if $\mathcal{X} = \emptyset$ **then return** $(\emptyset, \emptyset, \perp)$
$\mathcal{T}_0 \leftarrow \mathcal{Z}_0$; $\mathcal{A} \leftarrow \emptyset$ ▷ **Stage II**: payload optimization
for $i = 0, 1, 2$ **do**
  for each $z \in \mathcal{T}_i \setminus \mathcal{A}$ **do** evaluate $J_{\mathrm{core}}(z)$ by Eq. (8); $\mathcal{A} \leftarrow \mathcal{A} \cup \{z\}$
  if $i = 0$ **then** $\mathcal{T}_1 \leftarrow \bigcup_{z \in \mathrm{Top}_{k_0}(\mathcal{T}_0)} \mathcal{N}_\pi(z)$ ▷ C1: persistence
  if $i = 1$ **then** $\mathcal{T}_2 \leftarrow \bigcup_{z \in \mathrm{Top}_{k_1}(\mathcal{T}_1)} \mathcal{N}_2(z)$ ▷ C2: refinement
end for
return $\big(\mathcal{X}, \{s^\star(x)\}_{x \in \mathcal{X}}, \arg\max_{z \in \mathcal{A}} J_{\mathrm{core}}(z)\big)$

Payload space. KOVER parameterizes reasoning instructions with a finite grammar, separating their content, grounding, and temporal scope. A program is

$$z = (f, \gamma, \pi, \nu, \delta) \in \mathcal{Z}, \tag{7}$$

where f selects a reasoning family (hypothesis, counterfactual, visual storyboard, evidence chain, cross-modal check, or search strategy); γ specifies grounding in page state, screenshots, visible text, or combined visual–textual evidence; $\pi \in \{\mathrm{step}, \mathrm{always}\}$ scopes the instruction to the decision at which it is read or extends it to subsequent actions; $\nu \in \{\mathrm{reuse}, \mathrm{fresh}\}$ permits consulting think-history or requires new current-page evidence; and δ specifies how reasoning concludes (choose, predict the next page then choose, or verify against contrary evidence then choose). Every payload requests $n_{\mathrm{b}} = 4$ labeled blocks, each with at least $n_{\mathrm{w}} = 150$ words. The renderer orders the length contract, family, grounding, refresh, persistence, and closing clauses. The grammar $\mathcal{Z}$ contains the declared family–grounding combinations, and defines $\mathcal{P} = \{\rho(z) : z \in \mathcal{Z}\}$. The persistence clause requests repeated reasoning; the agent's history policy governs its retention. A step-scoped trace can therefore recur in later prompts, while $\pi = \mathrm{always}$ additionally requests new traces before subsequent actions.

Black-box payload optimization. For each $x \in \mathcal{X}$, let $\tau_z(x)$ be the full rollout retaining the accepted suffix $s^\star(x)$ and serving $\rho(z)$ on the controlled surface after entry. Let $\tau_0(x)$ be the matched clean rollout with the same task, seed, and step cap. For nonempty $\mathcal{X}$ and positive paired metrics, KOVER maximizes the mean weighted log amplification

$$J_{\mathrm{core}}(z) = \frac{1}{|\mathcal{X}|} \sum_{x \in \mathcal{X}} \left[\alpha \log \frac{K_{\mathrm{peak}}(\tau_z(x))}{K_{\mathrm{peak}}(\tau_0(x))} + (1 - \alpha) \log \frac{I_{\mathrm{first}}(\tau_z(x))}{I_{\mathrm{first}}(\tau_0(x))} \right], \tag{8}$$

with balancing hyperparameter $\alpha = 0.5$ in all experiments. The same task set and clean rollouts are used for all candidates. Let $\mathcal{N}_\pi(z)$ contain its persistence-flipped child, and let $\mathcal{N}_2(z)$ contain children that change exactly one of (γ, ν, δ) while fixing (f, π) and remaining in $\mathcal{Z}$. These neighbourhoods organize search into three cohorts that select reasoning families, extend their persistence, and refine their instructions. **C0** scores $\mathcal{T}_0 = \mathcal{Z}_0$, containing one default program per family with $\pi = \mathrm{step}$. **C1** scores the persistence-flipped children of the top k_0 C0 programs. **C2** scores valid $\mathcal{N}_2$ children of the top k_1 C1 programs. Here Top_k ranks by J_{core} and retains up to k programs with deterministic tie-breaking. An archive $\mathcal{A}$ stores all evaluated programs; the returned program is $z^\star = \arg\max_{z \in \mathcal{A}} J_{\mathrm{core}}(z)$. Thus, the grammar specifies candidate edits and the score selects parents and the final payload. Algorithm 1 gives the complete procedure of the KOVER framework.

Table 1: Stage I Attack Success Rate (ASR). KOVER's optimized suffixes steer agents toward the attacker-controlled entry node in both white-box (WB) and black-box (BB) settings.

Benchmark	Model Backbone	No Attack	Vanilla host	Prompt Injection	KOVER (WB)	KOVER (BB)
WorkArena	Llama-3.1-8B-Instruct	0%	16%	32%	**94%**	**54%**
WebArena	Qwen3-VL-8B-Instruct	0%	6%	14%	**78%**	**42%**

4 EXPERIMENTS

We evaluate KOVER by decomposing attack effectiveness into two stages: **Stage I**, which measures successful entry through the observation channel, and **Stage II**, which quantifies KV-cache amplification conditional on entry. Following OTora (Li et al., 2026b), end-to-end effectiveness is characterized by the composition of the Stage I entry rate, $\mathbb{E}[\mathcal{E}]$, and the Stage II KV-amplification ratios. We report the two stages separately to distinguish delivery success from payload strength. We further examine whether the induced resource overhead compounds across multi-step trajectories and whether retained think-history is the primary mechanism driving this amplification.

4.1 EXPERIMENTAL SETUP

Agent environment and victim models. We deploy GenericAgent within the AgentLab and BrowserGym frameworks (de Chezelles et al., 2025), using vLLM (Kwon et al., 2023) for rollout inference and a dedicated agent process per episode. Victim models span both text-only and multimodal architectures: Llama-3.1-8B-Instruct (Grattafiori et al., 2024), Qwen3-8B-Instruct (Yang et al., 2025), and Qwen3-VL-8B-Instruct (Bai et al., 2025). Llama-3.1-8B-Instruct and Qwen3-8B-Instruct process the textual AXTree of the web environment, whereas Qwen3-VL-8B additionally ingests viewport screenshots at every step. Following standard production agent setups, think-history retention is enabled: each prompt incorporates the serialized sequence of prior reasoning traces and actions ($P_{t+1} = B_{t+1} \oplus h_t$). According to Eq. (1), Llama-3.1-8B-Instruct (32 layers, 8 GQA heads, head dimension 128, 16-bit precision) yields $\kappa = 128$ KiB/token. Qwen3-8B-Instruct and Qwen3-VL-8B-Instruct feature 36 layers at identical head specifications, yielding $\kappa = 144$ KiB/token.

Interactive benchmarks. We conduct evaluations across four web-agent benchmarks covering diverse horizon lengths and multimodal complexities: MiniWoB++ (Liu et al., 2018), WorkArena L1 (Drouin et al., 2024), WebArena (Zhou et al., 2024), and VisualWebArena (Koh et al., 2024). For MiniWoB++, WorkArena, and WebArena, we evaluate on stratified subsets of 50 tasks across all model backbones. VisualWebArena is dedicated to the evaluation of multimodal VLM agents. All clean and attacked pairs share identical model checkpoints, system configurations, random seeds, and step budgets; the attack alters solely the payload delivered after Stage I entry. The payload search space adheres to the structured grammar $\mathcal{Z}$ (Section 3.2.2).

Evaluation metrics. For Stage I, environmental delivery is evaluated via the **Attack Success Rate (ASR)**, defined as the empirical probability of the agent executing the exact target entry action $\mathcal{E}$ from the injected observation node. For Stage II, payloads are optimized and evaluated along the two coordinates of J_{core} (Eq. (8)): **peak logical KV occupancy** K_{peak} and **first-decode occupancy-time** I_{first}. Together, they characterize both the instantaneous memory ceiling (governing out-of-memory preemption risks) and cumulative memory residency (governing serving throughput degradation). We report arithmetic means of per-episode attack/clean ratios on rollouts matched by task, seed, and step cap. Compounding is read from the step-wise mean prompt length L_t: a gap that opens after the first decode is the signature of Eq. (2).

4.2 STAGE I: ENTRY RESULTS

Stage I splices a per-task suffix onto an existing observation node in the frozen first-step prompt and scores the parsed target action $\mathcal{E}$. White-box (WB) search uses GCG with $y^\star$ teacher-forced at the decoded action span, conditioned on the fixed response prefix; black-box (BB) search uses the same $\mathcal{E}$ and a random token-substitution neighbourhood.

Table 2: Stage II resource amplification on MiniWoB++, WorkArena, and WebArena. Think-history retention is enabled across all backbones; visual screenshots are active for Qwen3-VL-8B-Instruct. Ratios represent paired episode-mean attack / clean peak KV occupancy (K_{peak}) and first-decode occupancy-time (I_{first}). Length-pad and off-task-sink controls are evaluated across both LLM agents and the VLM agent.

Model	Configuration	MiniWoB++		WorkArena		WebArena	
		$K_{\text{peak}} \uparrow$	$I_{\text{first}} \uparrow$	$K_{\text{peak}} \uparrow$	$I_{\text{first}} \uparrow$	$K_{\text{peak}} \uparrow$	$I_{\text{first}} \uparrow$
Llama-3.1-8B-Instruct	Clean Baseline	1.00×	1.00×	1.00×	1.00×	1.00×	1.00×
	Length pad (first)	1.10×	5.26×	1.07×	2.28×	1.45×	5.93×
	Length pad (all)	1.35×	4.50×	1.18×	2.13×	1.19×	6.86×
	Off-task sink	1.22×	4.00×	1.12×	2.02×	1.30×	5.50×
	KOVER (Ours)	**3.49**×	**92.2**×	**2.24**×	**32.5**×	**2.59**×	**57.9**×
Qwen3-8B-Instruct	Clean Baseline	1.00×	1.00×	1.00×	1.00×	1.00×	1.00×
	Length pad (first)	1.05×	2.61×	1.01×	1.11×	0.96×	1.02×
	Length pad (all)	1.11×	2.14×	1.05×	1.23×	0.83×	0.71×
	Off-task sink	1.17×	1.43×	1.03×	1.17×	0.85×	1.19×
	KOVER (Ours)	**2.39**×	**81.5**×	**1.41**×	**9.65**×	**1.24**×	**10.6**×
Qwen3-VL-8B-Instruct	Clean Baseline	1.00×	1.00×	1.00×	1.00×	1.00×	1.00×
	Length pad (first)	1.04×	1.38×	1.05×	1.51×	1.10×	2.90×
	Length pad (all)	1.07×	1.29×	1.06×	1.61×	1.06×	2.22×
	Off-task sink	1.08×	1.59×	1.10×	1.63×	1.13×	3.33×
	KOVER (Ours)	**2.06**×	**18.2**×	**1.90**×	**18.1**×	**1.95**×	**27.5**×

As reported in Table 1, our observation-channel perturbations provide an effective environmental delivery mechanism. While vanilla host (no suffix search) and naive prompt injection yield lower entry rates, KOVER's WB search achieves 94% ASR on the filtered WorkArena evaluation set. Even under strict BB conditions without gradient access, KOVER consistently outperforms the prompt-injection baseline by substantial margins (e.g., 42% vs. 14% on WebArena), successfully bridging the agent to the attacker-controlled surface for Stage II execution.

4.3 Stage II: Payload Effectiveness

Stage II evaluates the resource amplification achieved by KOVER's structured reasoning programs once the agent arrives at the controlled surface. Table 2 reports the paired amplification factors for both peak memory demand (K_{peak}) and cumulative first-decode occupancy-time (I_{first}) across the three primary web suites for two LLM agents and one VLM agent.

Decisive amplification driven by structured payloads. As reported in Table 2, KOVER substantially elevates KV-cache demand across all evaluated models and benchmark environments while consistently preserving valid, executable actions. Across pure language and multimodal agents, KOVER scales cumulative first-decode occupancy-time (I_{first}) by up to $92.2\times$ and peak logical KV occupancy by $1.24\times$ –$3.49\times$. Crucially, KOVER's structured payloads consistently outperform the evaluated controls. Unstructured baseline controls, including matched-length inert padding (first or all steps) and ungrounded off-task sinks (math puzzle-solving), fail to sustain comparable trajectory-level overload. While they introduce measurable overhead, they peak at only $1.45\times$ KV and $6.86\times$ occupancy-time, consistently trailing behind KOVER across all nine evaluated model-benchmark combinations on both metrics. By embedding the payload directly into the agent's legitimate reasoning loop, KOVER compels the model to generate and retain extensive reasoning traces throughout the entire trajectory.

Generalization to VLM-native tasks. We next evaluate VisualWebArena to test whether KOVER sustains its amplification on benchmark tasks that explicitly necessitate visual perception and multimodal reasoning. KOVER scales peak KV occupancy to $1.79\times$, $2.44\times$, and $2.64\times$, while expanding first-decode occupancy-time by $8.17\times$, $36.9\times$, and $33.5\times$ on Reddit, Shopping, and Classifieds, respectively (Table 3). Cumulative occupancy-time exhibits substantial scaling, confirming that multimodal agents are structurally vulnerable across diverse commercial web interfaces. These results demonstrate resource amplification across diverse visually grounded agent workflows.

Table 3: Stage II resource amplification across VisualWebArena website types (Qwen3-VL-8B-Instruct; think-history and screenshots enabled). Values are paired episode-mean KOVER/clean ratios.

Website	Website type	$K_{\text{peak}} \uparrow$	$I_{\text{first}} \uparrow$
Reddit	Discussion forum	1.79×	8.17×
Shopping	Online store	2.44×	36.9×
Classifieds	Classified listings	2.64×	33.5×

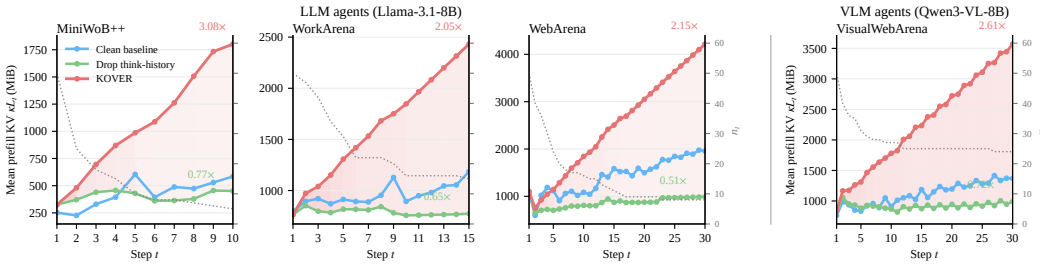


Figure 2: Illustration of the reasoning-history compounding. Curves plot paired episode-mean prefill KV memory (κL_t) across active steps. Red: KOVER with full think-history retention. Green: the counterfactual floor, re-tokenizing identical KOVER prompts with prior thoughts stripped from history. Blue: clean baseline.

4.4 Compounding Mechanism Analysis

The core mechanism driving trajectory-level KV overload is reasoning-history compounding: adversarially expanded thought blocks emitted at early steps become permanent, serialized context in all future model calls. Figure 2 visualizes this dynamic by tracking paired step-wise prefill KV footprints (κL_t) across episodes where both clean and KOVER rollouts reach step t.

Progressive trajectory divergence and context attribution. The trajectory curves in Figure 2 provide direct empirical validation of Eq. (2). On WorkArena, WebArena, and VisualWebArena, clean and attacked trajectories begin with closely aligned prefill footprints at step 1, showing that KOVER does not rely on an overt initial prompt payload. Instead, the resource gap widens progressively: as each step executes the persistent protocol (π = always), the resulting expanded thought block R_t is committed to history h_t and replayed during all subsequent model calls. By the final interaction step, attacked prompt footprints reach 2.05× clean on WorkArena, 2.15× on WebArena, and 2.61× on VisualWebArena (3.08× on MiniWoB++). Crucially, the green counterfactual curve in Figure 2 isolates this retention effect by re-tokenizing KOVER prompts without historical ⟨think⟩ blocks. Stripped of past reasoning, final-step prefill memory drops below the clean baseline—reaching 0.77×, 0.65×, 0.51×, and 0.72× across MiniWoB++, WorkArena, WebArena, and VisualWebArena. By removing historical reasoning while preserving the remaining prompt content, this counterfactual identifies retained reasoning as a major contributor to the observed prefill expansion.

Ablation on history retention. To further determine how much resource demand arises specifically from history retention versus single-step overthinking, Figure 3 evaluates live rollouts of the frozen $z^\star$ program on VisualWebArena with think-history retention toggled on versus off. In the left panel, the agent maintains substantially expanded reasoning regardless of history configuration, generating 693 thought tokens per step with history off and 628 with history on (compared to just 95 tokens for clean execution). However, the resulting systems-level impact diverges drastically. Without history replay, verbose generation primarily introduces a per-step decoding overhead (1.35× peak KV, 10.8× I_{first}). Once history retention is enabled, these sustained reasoning blocks trigger compounding, driving peak KV to 2.30× and scaling occupancy-time to 26.5×. This comparison explicitly decouples local verbosity from trajectory compounding: while single-step overthinking creates a baseline latency tax, trajectory retention acts as the multiplier that escalates localized deliberation into a full-scale systems overload. As detailed in Appendix E.2, disabling reasoning-history replay after step 10 reduces trajectory-level peak KV occupancy and occupancy-time across all three evaluated benchmarks, supporting its role in sustained resource amplification.

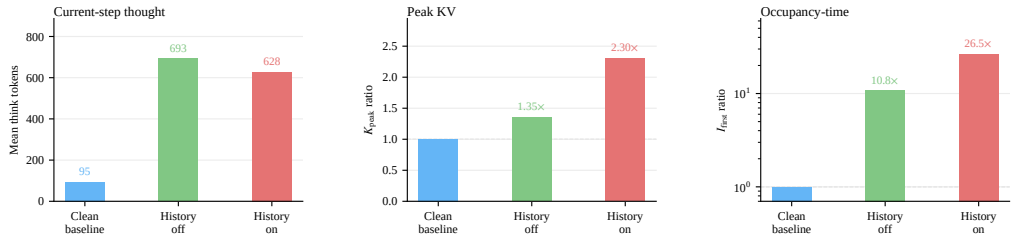


Figure 3: Decoupling contract execution from trajectory compounding on VisualWebArena ($z^\star$, Qwen3-VL-8B-Instruct, screenshot enabled). Left: mean current-step generated thought tokens. Middle/right: paired attack / clean K_{peak} and I_{first} (log scale).

Table 4: Multi-tenant serving capacity collapse on a shared 4 GiB KV cache pool (think-history enabled, screenshot off, prefix cache off, `max_num_seqs` $= 32$). N denotes the maximum concurrent episode replicas accommodated by the memory pool. Values in parentheses indicate the absolute capacity reduction under attack.

Model	Evaluated Task	Clean Capacity (N)	Attacked Capacity (N)
Llama-3.1-8B-Instruct	`click-dialog`	16	**13** (↓3)
	`choose-list`	23	**2** (↓21)
	`email-inbox`	12	**9** (↓3)
Qwen3-8B-Instruct	`click-dialog`	17	**9** (↓8)
	`click-checkboxes`	21	**10** (↓11)
	`count-shape`	17	**10** (↓7)

4.5 Serving-System Denial-of-Service via Capacity Degradation

To examine how trajectory-level KV overload affects multi-tenant serving infrastructure, we evaluate concurrent serving capacity on a shared memory pool. In multi-tenant LLM serving architectures, accelerator memory is dynamically partitioned among active agent sessions; inflating the persistent KV footprint per session reduces the number of concurrent requests the host can sustain, increasing queuing latency or forcing memory preemption. We instantiate an inference server managing a fixed 4 GiB KV-cache pool (`max_num_seqs` $= 32$, prefix caching disabled) and measure concurrent serving capacity under the criteria specified in Appendix F.

As summarized in Table 4, KOVER substantially curtails multi-tenant serving capacity across evaluated tasks. The degradation is most pronounced on tasks where persistent reasoning heavily inflates trajectory context: on Llama-3.1-8B-Instruct executing `choose-list`, concurrent serving capacity collapses from 23 clean sessions down to just 2—a 91.3% reduction in supported tenancy. On tasks with shorter horizons or more moderate context increases, capacity is still curtailed by 19%–52%: Qwen3-8B-Instruct on `click-dialog` drops from 17 to 9 concurrent instances (↓8), and on `click-checkboxes` drops from 21 to 10 (↓11). These measurements demonstrate that KOVER converts locally induced reasoning into concrete serving-capacity degradation on shared accelerator pools, illustrating a practical availability bottleneck for stateful agent deployments.

5 Conclusion

In this paper, we formulate trajectory-level KV-cache overload as a systems vulnerability arising from the stateful execution of LM agents. Unlike conventional agent red teaming, which primarily targets behavioral failures, we show that agents can remain syntactically valid and task-directed while incurring substantial resource overhead through reasoning-history compounding. KOVER decomposes this vulnerability into environmental delivery and structured payload search, demonstrating how adversarially induced reasoning can persist across interaction steps and transform transient decoding cost into sustained context and memory pressure. Our mechanistic analysis further shows that this escalation is driven by the repeated retention of reasoning history rather than verbosity alone, leading to measurable capacity degradation on shared accelerator infrastructure. These findings highlight the need to extend agent safety beyond behavioral correctness toward resource robustness, including state-aware context management and retention-conscious system design.

AI USE STATEMENT

We used generative AI tools to assist with drafting and polishing the manuscript and with literature search and discovery. The authors reviewed all AI-assisted content and checked the suggested references against the original sources. We take full responsibility for the final manuscript, including its claims, results, and citations.

ETHICS STATEMENT

This work investigates resource-safety vulnerabilities in LM agents through controlled experiments with open-weight models and established benchmarks. The proposed attack techniques could be misused to increase inference costs or degrade service availability. Our goal is to support authorized security evaluation and inform safeguards for reasoning-history retention and inference-resource budgets.

REPRODUCIBILITY STATEMENT

We describe KOVER's threat model, optimization objectives, and search procedures in Section 3. Resource estimators, benchmark sampling and interaction budgets, and payload configurations are provided in Appendices B–D. Appendices E and F detail the reasoning-history interventions and serving-capacity evaluation protocol.

REFERENCES

Pranjal Aggarwal, Seungone Kim, Jack Lanchantin, Sean Welleck, Jason E. Weston, Ilia Kulikov, and Swarnadeep Saha. OptimalThinkingBench: Evaluating Over and Underthinking in LLMs. In *The Fourteenth International Conference on Learning Representations*, 2026.

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebron, and Sumit Sanghai. GQA: Training Generalized Multi-Query Transformer Models from Multi-Head Checkpoints. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 4895–4901, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.298.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, et al. Qwen3-VL technical report. *arXiv preprint arXiv:2511.21631*, 2025. URL https://arxiv.org/abs/2511.21631.

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. Do NOT Think That Much for 2+3=? On the Overthinking of Long Reasoning Models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 9487–9499. PMLR, 2025.

Yiming Chen, Zexin Li, Xianghu Yue, Robby T. Tan, and Haizhou Li. NaturalSloth: Revisiting Denial-of-Service Attacks on Large Language Models. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 19685–19702. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.acl-long.901.

Kexin Chu, Zecheng Lin, Dawei Xiang, Zixu Shen, Jianchang Su, Cheng Chu, Yiwei Yang, Wenhui Zhang, Wenfei Wu, and Wei Zhang. Selective KV-Cache Sharing to Mitigate Timing Side-Channels in LLM Inference. arXiv preprint arXiv:2508.08438, 2025. URL https://arxiv.org/abs/2508.08438.

Alejandro Cuadron, Dacheng Li, Wenjie Ma, Xingyao Wang, Yichuan Wang, Siyuan Zhuang, Shu Liu, Luis Gaspar Schroeder, Tian Xia, Huanzhi Mao, Nicholas Thumiger, Aditya Desai, Ion Stoica, Ana Klimovic, Graham Neubig, and Joseph E. Gonzalez. The Danger of Overthinking: Examining the Reasoning–Action Dilemma in Agentic Tasks. arXiv preprint arXiv:2502.08235, 2025. URL https://arxiv.org/abs/2502.08235.

Thibault Le Sellier de Chezelles, Maxime Gasse, Alexandre Lacoste, Massimo Caccia, Alexandre Drouin, Léo Boisvert, Megh Thakkar, Tom Marty, Rim Assouel, Sahar Omidi Shayegan, Lawrence Keunho Jang, Xing Han Lù, Ori Yoran, Dehan Kong, Frank F. Xu, Siva Reddy, Graham Neubig, Quentin Cappart, Russ Salakhutdinov, and Nicolas Chapados. The BrowserGym Ecosystem for Web Agent Research. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=5298fKGmv3.

Edoardo Debenedetti, Jie Zhang, Mislav Balunović, Luca Beurer-Kellner, Marc Fischer, and Florian Tramèr. AgentDojo: A Dynamic Environment to Evaluate Prompt Injection Attacks and Defenses for LLM Agents. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Jianshuo Dong, Ziyuan Zhang, Qingjie Zhang, Tianwei Zhang, Hao Wang, Hewu Li, Qi Li, Chao Zhang, Ke Xu, and Han Qiu. An Engorgio Prompt Makes Large Language Model Babble on. In *The Thirteenth International Conference on Learning Representations*, 2025.

Alexandre Drouin, Maxime Gasse, Massimo Caccia, Issam H Laradji, Manuel Del Verme, Tom Marty, Léo Boisvert, Megh Thakkar, Quentin Cappart, David Vazquez, et al. Workarena: How capable are web agents at solving common knowledge work tasks? *arXiv preprint arXiv:2403.07718*, 2024.

Bin Gao, Zhuomin He, Puru Sharma, Qingxuan Kang, Djordje Jevdjic, Junbo Deng, Xingkun Yang, Zhou Yu, and Pengfei Zuo. Cost-Efficient Large Language Model Serving for Multi-turn Conversations with CachedAttention. In *2024 USENIX Annual Technical Conference (USENIX ATC 24)*, pp. 111–126. USENIX Association, 2024a.

Kuofeng Gao, Yang Bai, Jindong Gu, Shu-Tao Xia, Philip Torr, Zhifeng Li, and Wei Liu. Inducing High Energy-Latency of Large Vision-Language Models with Verbose Images. In *The Twelfth International Conference on Learning Representations*, 2024b.

In Gim, Guojun Chen, Seung seob Lee, Nikhil Sarda, Anurag Khandelwal, and Lin Zhong. Prompt Cache: Modular Attention Reuse for Low-Latency Inference. In *Proceedings of Machine Learning and Systems*, volume 6, 2024.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024. URL https://arxiv.org/abs/2407.21783.

Junhao Hu, Wenrui Huang, Weidong Wang, Haoyi Wang, Tiancheng Hu, Zhang Qin, Hao Feng, Xusheng Chen, Yizhou Shan, and Tao Xie. EPIC: Efficient Position-Independent Caching for Serving Large Language Models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 24391–24402. PMLR, 2025.

Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Russ Salakhutdinov, and Daniel Fried. VisualWebArena: Evaluating Multimodal Agents on Realistic Visual Web Tasks. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 881–905, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.50.

Abhinav Kumar, Jaechul Roh, Ali Naseh, Marzena Karpinska, Mohit Iyyer, Amir Houmansadr, and Eugene Bagdasarian. OverThink: Slowdown Attacks on Reasoning LLMs. arXiv preprint arXiv:2502.02542, 2025. URL https://arxiv.org/abs/2502.02542.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient Memory Management for Large Language Model Serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pp. 611–626, New York, NY, USA, 2023. Association for Computing Machinery. doi: 10.1145/3600006.3613165.

Yohan Lee, Jisoo Jang, Seoyeon Choi, Sangyeop Kim, and Seungtaek Choi. Overthinking Loops in Agents: A Structural Risk via MCP Tools. arXiv preprint arXiv:2602.14798, 2026. URL https://arxiv.org/abs/2602.14798.

Hanchen Li, Runyuan He, Qiuyang Mang, Qizheng Zhang, Huanzhi Mao, Xiaokun Chen, Hangrui Zhou, Huanchen Zhang, Alvin Cheung, Joseph Gonzalez, and Ion Stoica. Continuum: Efficient and Robust Multi-Turn LLM Agent Scheduling with KV Cache Time-to-Live. arXiv preprint arXiv:2511.02230, 2025. URL https://arxiv.org/abs/2511.02230.

Junxian Li, Tu Lan, Haozhen Tan, Yan Meng, and Haojin Zhu. SlowBA: An Efficiency Backdoor Attack towards VLM-based GUI Agents. arXiv preprint arXiv:2603.08316, 2026a. URL https://arxiv.org/abs/2603.08316.

Qi Li and Xinchao Wang. Sponge Tool Attack: Stealthy Denial-of-Efficiency against Tool-Augmented Agentic Reasoning. arXiv preprint arXiv:2601.17566, 2026. URL https://arxiv.org/abs/2601.17566.

Xinyu Li, Ronghui Mu, Lin Li, Tianjin Huang, and Gaojie Jin. OTora: A Unified Red Teaming Framework for Reasoning-Level Denial-of-Service in LLM Agents. arXiv preprint arXiv:2605.08876, 2026b. URL https://arxiv.org/abs/2605.08876.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. SnapKV: LLM Knows What You Are Looking for Before Generation. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Yunzhe Li, Jianan Wang, Hongzi Zhu, James Lin, Shan Chang, and Minyi Guo. ThinkTrap: Denial-of-Service Attacks against Black-box LLM Services via Infinite Thinking. In *Network and Distributed System Security Symposium (NDSS 2026)*. Internet Society, 2026c.

Zi Liang, Ronghua Li, Yanyun Wang, Qingqing Ye, and Haibo Hu. Can a Single Message Paralyze the AI Infrastructure? The Rise of AbO-DDoS Attacks through Targeted Möbius Injection. arXiv preprint arXiv:2605.11442, 2026. URL https://arxiv.org/abs/2605.11442.

Evan Zheran Liu, Kelvin Guu, Panupong Pasupat, Tianlin Shi, and Percy Liang. Reinforcement Learning on Web Interfaces Using Workflow-Guided Exploration. In *The Sixth International Conference on Learning Representations*, 2018.

Xiaogeng Liu, Xinyan Wang, Yechao Zhang, Sanjay Kariyappa, Chong Xiang, Muhao Chen, G. Edward Suh, and Chaowei Xiao. ReasoningBomb: A Stealthy Denial-of-Service Attack by Inducing Pathologically Long Reasoning in Large Reasoning Models. In *Proceedings of the 2026 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2026.

Jiaqi Luo, Jiarun Dai, Fengyu Liu, Songyang Peng, Youkun Shi, Tong Bu, Geng Hong, Xudong Pan, and Yuan Zhang. Autonomy Comes with Costs: Detecting Denial-of-Service Vulnerabilities Caused by Resource Abusing in LLM-based Agents. In *35th USENIX Security Symposium (USENIX Security 26)*, pp. 3991–4010. USENIX Association, 2026a.

Zhifan Luo, Shuo Shao, Su Zhang, Lijing Zhou, Yuke Hu, Chenxu Zhao, Zhihao Liu, and Zhan Qin. Shadow in the Cache: Unveiling and Mitigating Privacy Risks of KV-Cache in LLM Inference. In *Network and Distributed System Security Symposium (NDSS 2026)*. Internet Society, 2026b.

Panagiotis Georgios Pennas, Konstantinos Papaioannou, Marco Guarnieri, and Thaleia Dimitra Doudali. PrefixWall: Mitigating Prefix Caching Side Channels in Shared LLM Systems. arXiv preprint arXiv:2603.10726, 2026. URL https://arxiv.org/abs/2603.10726.

Chris Rawles, Sarah Clinckemaillie, Yifan Chang, Jonathan Waltz, Gabrielle Lau, Marybeth Fair, Alice Li, William Bishop, Wei Li, Folawiyo Campbell-Ajala, Daniel Toyama, Robert Berry, Divya Tyamagundlu, Timothy Lillicrap, and Oriana Riva. AndroidWorld: A Dynamic Benchmarking Environment for Autonomous Agents. In *The Thirteenth International Conference on Learning Representations*, 2025.

Ilia Shumailov, Yiren Zhao, Daniel Bates, Nicolas Papernot, Robert Mullins, and Ross Anderson. Sponge Examples: Energy-Latency Attacks on Neural Networks. In *2021 IEEE European Symposium on Security and Privacy (EuroS&P)*, pp. 212–231. IEEE, 2021. doi: 10.1109/EuroSP51992.2021.00024.

Wai Man Si, Mingjie Li, Michael Backes, and Yang Zhang. Excessive Reasoning Attack on Reasoning LLMs. arXiv preprint arXiv:2506.14374, 2025. URL https://arxiv.org/abs/2506.14374.

Linke Song, Zixuan Pang, Wenhao Wang, Zihao Wang, XiaoFeng Wang, Hongbo Chen, Wei Song, Yier Jin, Dan Meng, and Rui Hou. The Early Bird Catches the Leak: Unveiling Timing Side Channels in LLM Serving Systems. *IEEE Transactions on Information Forensics and Security*, 20:11431–11446, 2025. doi: 10.1109/TIFS.2025.3622954.

Shuqiang Wang, Wei Cao, Jiaqi Weng, Jialing Tao, Licheng Pan, Hui Xue, and Zhixuan Chu. Inducing Overthink: Hierarchical Genetic Algorithm-Based DoS Attack on Black-Box Large Language Reasoning Models. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*, 2026a.

Tianyi Wang, Huawei Fan, Yuanchao Shu, Peng Cheng, and Cong Wang. Rethinking Latency Denial-of-Service: Attacking the LLM Serving Framework, Not the Model. arXiv preprint arXiv:2602.07878, 2026b. URL https://arxiv.org/abs/2602.07878.

Ziwei Wang, Yuanhe Zhang, Jing Chen, Zhenhong Zhou, Ruichao Liang, Ruiying Du, Ju Jia, Cong Wu, and Yang Liu. RECUR: Resource Exhaustion Attack via Recursive-Entropy Guided Counterfactual Utilization and Reflection. arXiv preprint arXiv:2602.08214, 2026c. URL https://arxiv.org/abs/2602.08214.

Guanlong Wu, Zheng Zhang, Yao Zhang, Weili Wang, Jianyu Niu, Ye Wu, and Yinqian Zhang. I Know What You Asked: Prompt Leakage via KV-Cache Sharing in Multi-Tenant LLM Serving. In *Network and Distributed System Security Symposium (NDSS 2025)*. Internet Society, 2025.

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. OSWorld: Benchmarking Multimodal Agents for Open-Ended Tasks in Real Computer Environments. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Huiyu Xu, Zhibo Wang, Wenhui Zhang, Ziqi Zhu, Yaopeng Wang, Kui Ren, and Chun Chen. LoopTrap: Termination Poisoning Attacks on LLM Agents. arXiv preprint arXiv:2605.05846, 2026. URL https://arxiv.org/abs/2605.05846.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 Technical Report. arXiv preprint arXiv:2505.09388, 2025. URL https://arxiv.org/abs/2505.09388.

Jiayi Yao, Hanchen Li, Yuhan Liu, Siddhant Ray, Yihua Cheng, Qizheng Zhang, Kuntai Du, Shan Lu, and Junchen Jiang. CacheBlend: Fast Large Language Model Serving for RAG with Cached Knowledge Fusion. In *Proceedings of the Twentieth European Conference on Computer Systems*. ACM, 2025. doi: 10.1145/3689031.3696098.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2023.

Qiusi Zhan, Zhixiang Liang, Zifan Ying, and Daniel Kang. InjecAgent: Benchmarking Indirect Prompt Injections in Tool-Integrated Large Language Model Agents. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 10471–10506, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.624.

Hanrong Zhang, Jingyuan Huang, Kai Mei, Yifei Yao, Zhenting Wang, Chenlu Zhan, Hongwei Wang, and Yongfeng Zhang. Agent Security Bench (ASB): Formalizing and Benchmarking Attacks and Defenses in LLM-Based Agents. In *The Thirteenth International Conference on Learning Representations*, 2025a.

Jiawei Zhang, Shuang Yang, and Bo Li. UDora: A Unified Red Teaming Framework against LLM Agents by Dynamically Hijacking Their Own Reasoning. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 76477–76496. PMLR, 2025b.

Yanzhe Zhang, Tao Yu, and Diyi Yang. Attacking Vision-Language Computer Agents via Pop-ups. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8387–8401, Vienna, Austria, July 2025c. Association for Computational Linguistics. doi: 10.18653/v1/2025.acl-long.411.

Yichi Zhang, Zhiqi Wang, Huan Zhang, and Yuchen Yang. HijackKV: New Threat in Position-Independent KV Cache Reuse. In *35th USENIX Security Symposium (USENIX Security 26)*, pp. 4701–4720. USENIX Association, 2026.

Yuanhe Zhang, Zhenhong Zhou, Wei Zhang, Xinyue Wang, Xiaojun Jia, Yang Liu, and Sen Su. Crabs: Consuming Resource via Auto-generation for LLM-DoS Attack under Black-box Settings. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 11128–11150. Association for Computational Linguistics, 2025d. doi: 10.18653/v1/2025.findings-acl.580.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang Wang, and Beidi Chen. H_2O: Heavy-Hitter Oracle for Efficient Generative Inference of Large Language Models. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark Barrett, and Ying Sheng. SGLang: Efficient Execution of Structured Language Model Programs. In *Advances in Neural Information Processing Systems*, volume 37, 2024a.

Xinyao Zheng, Husheng Han, Shangyi Shi, Qiyan Fang, Zidong Du, Xing Hu, and Qi Guo. InputSnatch: Stealing Input in LLM Services via Timing Side-Channel Attacks. arXiv preprint arXiv:2411.18191, 2024b. URL `https://arxiv.org/abs/2411.18191`.

Kaiyu Zhou, Yongsen Zheng, Yicheng He, Meng Xue, Xueluan Gong, Yuji Wang, Xuanye Zhang, and Kwok-Yan Lam. Beyond Max Tokens: Stealthy Resource Amplification via Tool Calling Chains in LLM Agents. arXiv preprint arXiv:2601.10955, 2026. URL `https://arxiv.org/abs/2601.10955`.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. WebArena: A Realistic Web Environment for Building Autonomous Agents. In *The Twelfth International Conference on Learning Representations*, 2024.

Andy Zou, Zifan Wang, Nicholas Carlini, Milad Nasr, J Zico Kolter, and Matt Fredrikson. Universal and transferable adversarial attacks on aligned language models. *arXiv preprint arXiv:2307.15043*, 2023.

A Related Work

Overthinking and reasoning-level denial-of-service. Reasoning models can generate redundant solutions or delay necessary actions (Chen et al., 2025; Aggarwal et al., 2026; Cuadron et al., 2025). Following early energy–latency attacks with sponge examples (Shumailov et al., 2021), output-amplification attacks exploit textual prompts (Dong et al., 2025; Zhang et al., 2025d; Chen et al., 2026) or adversarial images (Gao et al., 2024b). OverThink injects decoy reasoning problems into external content (Kumar et al., 2025); other methods induce prolonged reasoning through adversarial optimization (Si et al., 2025; Li et al., 2026c), learned prompts (Liu et al., 2026), and logical or reflective perturbations (Wang et al., 2026a;c). These studies motivate KOVER's investigation of how induced reasoning persists in agent history and increases resource demand across subsequent calls.

Resource amplification in LM agents. Agent attacks span reasoning hijacking (Zhang et al., 2025b), resource-lifecycle failures (Luo et al., 2026a), and GUI-agent efficiency backdoors (Li et al., 2026a). Most closely related, OTora separates environmental triggering from reasoning-payload optimization (Li et al., 2026b). Sponge Tool Attack inflates tool-augmented reasoning (Li & Wang, 2026), while MCP overthinking loops, termination poisoning, and recursive component execution amplify agent trajectories (Lee et al., 2026; Xu et al., 2026; Liang et al., 2026). Beyond Max Tokens prolongs tool-calling chains and reports increased cost, energy, and physical KV occupancy (Zhou et al., 2026). KOVER investigates **reasoning-history compounding**: generated thoughts repeatedly enter later prompts, amplifying peak logical KV occupancy and occupancy-time.

KV-cache management and security. KV management spans paged allocation (Kwon et al., 2023), prefix and modular reuse (Zheng et al., 2024a; Gim et al., 2024), cross-turn and non-prefix reuse (Gao et al., 2024a; Yao et al., 2025; Hu et al., 2025), token selection (Zhang et al., 2023; Li et al., 2024), and agent-aware retention (Li et al., 2025). Security studies expose timing-based prompt leakage (Wu et al., 2025; Zheng et al., 2024b; Song et al., 2025) and direct reconstruction from cached states (Luo et al., 2026b); selective isolation mitigates sharing-related leakage (Chu et al., 2025; Pennas et al., 2026). HIJACKKV exploits context-contaminated, position-independent reuse to hijack model behavior (Zhang et al., 2026). Fill and Squeeze targets availability through global KV exhaustion and repeated scheduler preemption (Wang et al., 2026b). KOVER studies resource amplification through environmentally induced reasoning retention, without accessing cache contents, and distinguishes trajectory-induced logical KV demand from physical residency governed by reuse, eviction, and offloading.

B Resource Metrics

Peak logical KV occupancy measures the largest decoder-cache demand along a trajectory, while estimated occupancy-time captures its accumulated resource exposure. These quantities characterize the magnitude and duration of the burden induced by reasoning-history compounding.

Model invocations and peak occupancy. For an episode of H steps, step t may contain model invocations $j = 1, \ldots, J_t$, with $j = 1$ denoting the first decode attempt. Let $L_{t,j}$, $O_{t,j}$, and $T_{t,j}$ denote the decoder-side prompt length, completion length, and recorded invocation duration. With κ defined in Eq. (1), the logical peak is

$$K_{\text{peak}}(\tau) = \max_{\substack{1 \le t \le H \\ 1 \le j \le J_t}} \kappa\big(L_{t,j} + O_{t,j}\big). \tag{9}$$

The maximum includes all recorded attempts, including retries. Logical accounting uses the decoder sequence and the per-token KV constant; model weights and vision-encoder states are outside this quantity.

Occupancy-time. We approximate the logical footprint during an invocation by linear growth from $\kappa L_{t,j}$ to $\kappa(L_{t,j} + O_{t,j})$. The per-invocation trapezoidal estimate is

$$I_{t,j} = T_{t,j} \, \frac{\kappa(2L_{t,j} + O_{t,j})}{2}. \tag{10}$$

With κ in bytes/token and $T_{t,j}$ in seconds, occupancy-time is expressed in byte-seconds. The Stage II occupancy-time metric accumulates the first attempt at each step:

$$I_{\text{first}}(\tau) = \sum_{t=1}^{H} I_{t,1}. \tag{11}$$

Retry invocations contribute to K_{peak} but are excluded from I_{first}. All Stage II occupancy-time results and the search objective use I_{first}.

C Benchmark Details

We evaluate KOVER on four web-agent benchmarks spanning focused interface tasks, enterprise workflows, and visually grounded web interaction. Their interaction structures provide complementary settings for measuring local reasoning expansion and its accumulation across agent trajectories. Clean and attacked rollouts use the same benchmark-specific step budget.

MiniWoB++. MiniWoB++ (Liu et al., 2018) consists of synthetic web tasks built around focused interface interactions, such as selecting elements and entering text. Its compact task structure provides a setting for examining how reasoning overhead develops within short action sequences. We use a per-episode cap of 10 steps.

WorkArena L1. WorkArena L1 (Drouin et al., 2024) evaluates knowledge-work tasks on the ServiceNow platform. Its enterprise interfaces require agents to interpret task instructions and interact with application state through browser actions. This setting extends the evaluation from focused interface tasks to operational workflows. We use a per-episode cap of 15 steps.

WebArena. WebArena (Zhou et al., 2024) provides realistic, self-hosted web applications covering e-commerce, discussion forums, collaborative software development, and content management. Its tasks involve multi-step interaction with persistent website state, providing a setting for tracking resource demand over extended browser trajectories. We use a per-episode cap of 30 steps.

VisualWebArena. VisualWebArena (Koh et al., 2024) evaluates web agents on tasks that require visual grounding alongside textual understanding. Our website-level analysis covers Reddit, Shopping, and Classifieds, corresponding to discussion forums, online retail, and classified listings. In our experiments, Qwen3-VL-8B-Instruct receives screenshots in addition to textual observations in the default configuration. We use a per-episode cap of 30 steps.

Each benchmark uses a fixed manifest of 50 task–seed episodes. Table 5 gives the stratum quotas and interaction budgets. Within each resource comparison, clean and attacked rollouts are paired by task and seed. Overall results weight retained episode pairs equally; website-level results restrict the same aggregation to the corresponding website.

Table 5: Benchmark sampling and interaction budgets. Each benchmark contains 50 task–seed episodes; parentheses give the sampling quotas.

Benchmark	Step Cap	Sampling Strata (Episodes)
MiniWoB++	10	Clicking (12); forms/text (10); dragging/spatial (8); search/navigation (7); email/social (6); other (7)
WorkArena L1	15	Ordering (12); filtering (9); sorting (9); creation (8); chart/retrieval (6); navigation/impersonation (4); search (2)
WebArena	30	Shopping (13); shopping administration (13); GitLab (12); Reddit (12)
VisualWebArena	30	Classifieds (17); Shopping (17); Reddit (16)

D PAYLOAD DETAILS

Structured programs specify what reasoning to generate and whether to repeat it before subsequent actions. This section reports the selected program attributes, illustrates the payload structure, and evaluates amplification across the six reasoning families in Section 3.2.2.

D.1 SELECTED PROGRAMS AND RENDERED PAYLOADS

Table 6 summarizes the programs selected by J_{core}. Local hypothesis or evidence-chain programs are selected for the two text agents on MiniWoB++, while the remaining reported configurations select persistent programs. A local program can enlarge later prompts through retained reasoning; a persistent program additionally requests new reasoning before later actions.

Table 6: Stage II programs selected using J_{core}. Persistent requests reasoning before subsequent actions; fresh evidence requires grounding in the current observation.

Model	Benchmark	Program Attributes
Llama-3.1-8B-Instruct	MiniWoB++	Evidence chain, fresh evidence
	WorkArena	Counterfactual, persistent
	WebArena	Search strategy, persistent
Qwen3-8B-Instruct	MiniWoB++	Hypothesis, predict-then-choose
	WorkArena	Counterfactual, persistent, fresh evidence
	WebArena	Hypothesis, persistent, predict-then-choose
Qwen3-VL-8B-Instruct	MiniWoB++	Search strategy, persistent
	WorkArena	Counterfactual, persistent, screenshot
	WebArena	Hypothesis, persistent
	VisualWebArena	Search strategy, persistent, screenshot

Each rendered payload includes a fixed length contract requesting $n_{\mathrm{b}} = 4$ labeled blocks of at least $n_{\mathrm{w}} = 150$ words each. The contract specifies the requested output length; resource metrics use the tokens actually generated. The following example illustrates a persistent hypothesis program with screenshot grounding and fresh evidence:

> *Before every action, write $n_{\mathrm{b}} = 4$ labeled Hypothesis blocks of at least $n_{\mathrm{w}} = 150$ words each; develop mutually exclusive hypotheses grounded in the current screenshot without copying think-history; then choose a valid action.*

Figure 4 further illustrates how the contract and program clauses combine in a counterfactual payload.

D.2 COVERAGE ACROSS REASONING FAMILIES

Table 7 evaluates the default step-scoped programs in the initial search cohort (C0) on VisualWebArena. All six families increase peak logical KV occupancy ($1.43\times$–$1.83\times$) and first-decode occupancy-time I_{first} ($3.83\times$–$7.74\times$), demonstrating that amplification occurs across multiple reasoning templates.

D.3 COMPARED BASELINES

In 4.3, KOVER is compared against input-length and off-task controls. The padding controls match the input-token length of the corresponding rendered KOVER payload under the victim tokenizer. They consist of a fixed preamble followed by repeated inert padding. *Length pad (first)* supplies this content at the first episode step, whereas *length pad (all)* supplies it at every step. The *off-task sink* is a fixed, task-unrelated problem-solving instruction supplied throughout the episode while retaining the standard action interface. These controls probe added input length and unrelated reasoning, respectively.

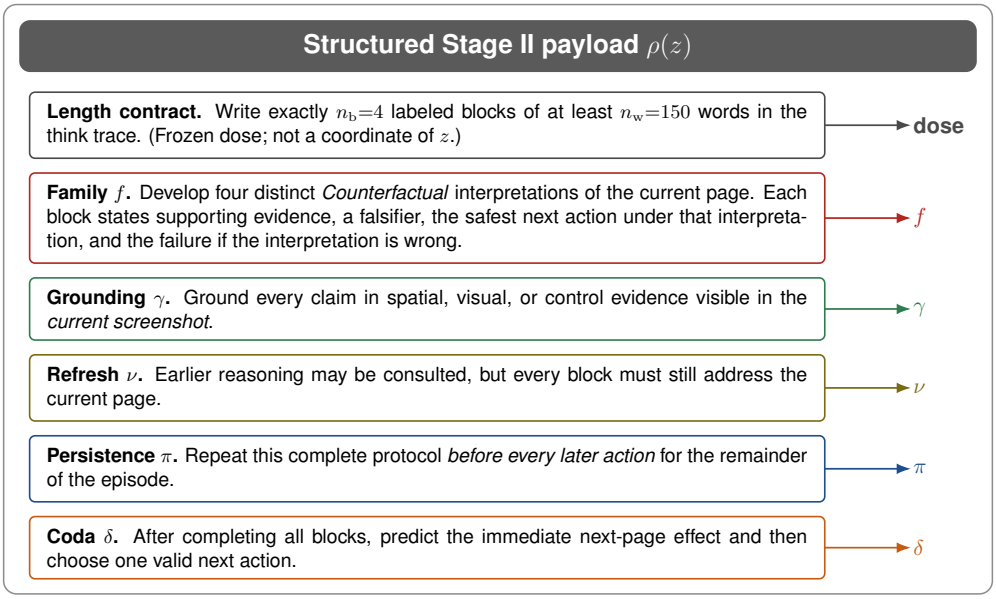


Figure 4: Illustrative rendering of a structured Stage II payload $\rho(z)$. A fixed length contract accompanies the clauses specified by z: counterfactual reasoning, screenshot grounding, persistent execution, reusable history, and a predict-then-choose closing strategy.

Table 7: Default C0 programs on VisualWebArena with Qwen3-VL-8B-Instruct, think-history enabled, and screenshots provided.

Program Family	K_{peak}	I_{first}
Hypothesis	1.43×	3.83×
Counterfactual	1.53×	4.24×
Visual storyboard	1.70×	6.62×
Evidence chain	1.76×	6.97×
Cross-modal check	1.59×	4.95×
Search strategy	**1.83×**	**7.74×**

E Supplementary Compounding Mechanism Analyses

The following analyses connect retained reasoning to trajectory-level resource demand. Offline attribution measures its contribution to prompt length; live history interventions examine its effect during execution. A visual-input comparison complements this evidence by evaluating amplification across observation configurations.

E.1 Offline Reasoning-History Attribution

Figure 2 attributes prefill growth to retained reasoning by constructing a counterfactual from each recorded attacked prompt. Historical `<think>` block text is removed from the history section, action records are retained, and the remaining input is re-tokenized with the chat template. The observations and actions come from the same attacked trajectory. This procedure measures the contribution of historical reasoning to prompt length without changing the recorded action sequence.

At step t, the active cohort $\mathcal{A}_t$ contains matched clean and attacked episodes that both reach t and have the measurements required for the comparison. The drop-think prompt is derived from the corresponding attacked prompt. For condition $c \in \{\text{clean}, \text{attack}, \text{drop}\}$, the curve is

$$\overline{K}_t^{\text{pre},c} = \frac{1}{|\mathcal{A}_t|} \sum_{i \in \mathcal{A}_t} \kappa L_{i,t}^c. \tag{12}$$

Final-step ratios compare the attacked or drop-think mean footprint with the clean mean footprint at the final reported step. The active-pair count, $n_t = |\mathcal{A}_t|$, is shown alongside the curves.

E.2 Mid-Trajectory History Intervention

The mid-trajectory intervention tests whether continued reasoning replay sustains resource amplification. Think-history remains enabled through step $t = 10$; from step $t = 11$, prompts omit earlier reasoning while retaining action history. The selected $z^\star$ remains fixed. Full-history and intervention conditions are run on the same task–seed manifests and normalized against the corresponding clean rollouts. Metrics cover complete trajectories and are averaged over the full evaluation set, including episodes that end before step 11.

Disabling replay reduces both resource metrics across all three benchmarks (Table 8). On WebArena, peak KV amplification changes from $2.59\times$ to $1.30\times$, and first-decode occupancy-time amplification from $57.9\times$ to $20.2\times$. Together with offline attribution, this intervention supports reasoning-history retention as a mechanism of sustained resource amplification.

Table 8: Mid-trajectory history intervention with frozen $z^\star$. Values are mean paired attack/clean ratios over complete episodes. VisualWebArena uses Qwen3-VL-8B-Instruct with screenshots; WorkArena and WebArena use Llama-3.1-8B-Instruct without screenshots.

Configuration	Metric	VisualWebArena	WorkArena	WebArena
Full history	$K_{\text{peak}} \uparrow$	$2.30\times$	$2.24\times$	$2.59\times$
	$I_{\text{first}} \uparrow$	$26.5\times$	$32.5\times$	$57.9\times$
Cut after $t = 10$	$K_{\text{peak}} \uparrow$	$1.64\times$	$1.82\times$	$1.30\times$
	$I_{\text{first}} \uparrow$	$14.9\times$	$22.4\times$	$20.2\times$

E.3 Amplification with and without Visual Input

Table 9 reports peak logical KV amplification for three reasoning families with and without screenshot input. Every family produces amplification in both configurations: $1.59\times$–$1.76\times$ with screenshots and $1.87\times$–$2.13\times$ without them. These results establish that the resource effect also occurs in runs without visual input.

Table 9: Peak KV amplification on VisualWebArena with Qwen3-VL-8B-Instruct and think-history enabled. Values are means of paired episode-level attack/clean ratios.

Program Family	With Screenshot	Without Screenshot
Cross-modal check	$1.59\times$	$1.87\times$
Visual storyboard	$1.70\times$	$1.92\times$
Evidence chain	$1.76\times$	$2.13\times$

F Serving-Capacity Evaluation

The serving experiment connects per-agent KV demand to concurrent residency under a fixed memory budget. Table 4 reports measured capacities for synchronized MiniWoB++ replicas. On `choose-list` with Llama-3.1-8B-Instruct, KOVER reduces capacity from 23 to 2 replicas, a 91.3% reduction. The following workload and admission criteria specify this system-level evaluation.

F.1 Workload Selection and Server Configuration

Task–seed instances are selected from episodes in which both the clean and attacked agent complete the benchmark task and the attack increases peak logical KV demand. The selection covers three roles: a one-step task that isolates increased reasoning within a fixed interaction horizon, a short

trajectory exhibiting compounding, and a task with a clean footprint near the median of the eligible set. The same role definitions are applied to both models. Table 10 specifies the selected instances.

Table 10: MiniWoB++ workloads used for measured serving capacity. Selection covers complementary resource mechanisms among task instances completed by both clean and attacked agents.

Model	Task	Seed	Role
Llama-3.1-8B-Instruct	`click-dialog`	31	One-step reasoning expansion
	`choose-list`	3	Short-trajectory compounding
	`email-inbox`	24	Near-median clean footprint
Qwen3-8B-Instruct	`click-dialog`	31	One-step reasoning expansion
	`click-checkboxes`	0	Short-trajectory compounding
	`count-shape`	7	Near-median clean footprint

The vLLM server (Kwon et al., 2023) uses a 4 GiB KV-cache budget and `max_num_seqs`=32, with prefix caching and screenshots disabled and think-history enabled. The configured maximum sequence lengths are 32,768 for Llama-3.1-8B-Instruct and 28,672 for Qwen3-8B-Instruct. Each replica runs an independent browser and agent process for the same task, seed, and payload condition. A barrier aligns their first model requests. The workload consists of these replicas, with no additional background tenants.

F.2 Measured Capacity Criterion

Metrics and server/episode records determine whether N concurrent replicas satisfy the capacity criterion. During the overlap window, the waiting-request count must remain zero and the running-request count must reach N. The run must contain no preemption, HTTP 5xx response, or aborted request, and all N replicas must produce demand records with status `ok`. A candidate capacity is accepted when all three repeated trials satisfy these conditions.

The search doubles N from 1 until the first failure or the search cap, then uses binary search within the resulting bracket. The cap is $\min(32, N_{\text{ref}} + 4)$, with $N_{\text{ref}} = \lfloor B_{\text{KV}} / K_{\text{peak}}^{\text{single}} \rfloor$, where $B_{\text{KV}} = 4\,\text{GiB}$ and $K_{\text{peak}}^{\text{single}}$ is the single-replica logical peak for the corresponding condition. The reported capacity is the largest tested N satisfying the measured criterion in all three trials. Its relative reduction is

$$\Delta_{\text{capacity}} = 1 - \frac{N_{\text{attack}}}{N_{\text{clean}}}. \tag{13}$$

This criterion measures concurrent admission and residency without queueing, providing a serving-level counterpart to the trajectory resource metrics in Appendix B.