Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

ChronosAttack: Adversarial Tool Scheduling Attacks on LLM Agents

Masih Moafi
Isfahan University of Technology
Arash Vashagh
arash.vashagh@unb.ca
University of New Brunswick https://orcid.org/0009-0001-4006-6683

Research Article

Keywords: LLM agents, tool-use security, adversarial scheduling, order sensitivity, asynchronous agents, prompt injection

Posted Date: October 30th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-10854515/v2

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: The authors declare no competing interests.

ChronosAttack: Adversarial Tool Scheduling Attacks on LLM Agents

Masih Moafi† and Arash Vashagh*‡
† Department of Electrical and Computer Engineering, Isfahan University of Technology, Isfahan, Iran
‡ Faculty of Computer Science, University of New Brunswick, Fredericton, NB E3B 5A3, Canada
m.moafi@el.iut.ac.ir, arash.vashagh@unb.ca

***Abstract*—We introduce ChronosAttack, an attack on large language model (LLM) agents that changes only when authentic tool results arrive. When an agent invokes several tools in parallel, the runtime may pass each result to the model as soon as it arrives. Tool completion times then determine the order in which the model reads the evidence. ChronosAttack exploits this behavior by delaying results without changing their contents or adding or removing observations. Checks that inspect output content alone therefore have no altered payload to flag. We formalize this delay-only threat model and consider a fixed attacker, which chooses a delivery order in advance, and an adaptive attacker, which tests orders using observed decisions. We distinguish *order sensitivity*, the dependence of a decision on evidence order, from *steering*, the attacker's ability to move a decision toward a chosen option. Using controlled replay on synthetic decision tasks, we evaluate both properties and compare defenses that change how agents receive or combine tool results. We assess these defenses by their resistance to steering and their effect on decisions in the absence of an attack.**

***Index Terms*—LLM agents, tool-use security, adversarial scheduling, order sensitivity, asynchronous agents, prompt injection.**

I. Introduction

Large language models (LLMs) are increasingly used as agents that call external tools, read what the tools return, and act on the results [1], [2]. The model requests the calls. The receiving infrastructure executes the tools, collects their outputs, and assembles the next prompt. Early frameworks ran one call at a time [1], whereas later systems run several calls at once to reduce latency [3] and either return the outputs in call order or pass each output to the model as soon as its tool finishes [4]. When outputs arrive as tools finish, the infrastructure determines their order in the prompt.

Previous research on the security of LLM agents has primarily focused on the content they receive. Indirect prompt-injection attacks embed malicious instructions in content retrieved by agents [5]. Existing benchmarks evaluate how frequently malicious tool outputs can manipulate an agent's behavior [6], while defense mechanisms aim to detect or isolate such content [7]. Separately, studies of order effects show that rearranging information within a prompt can change an LLM's response [8]–[10]. These two lines of work leave a security question unanswered: when an asynchronous agent receives several authentic tool results, can an adversary steer its decision by controlling only when those results arrive? In this setting, the evidence is neither forged nor edited; its order is determined by the runtime's completion schedule.

ChronosAttack, introduced in our earlier preprint [11], makes this scheduling behavior an explicit attack surface. An attacker delays tool results to change their delivery order while preserving every result's content. The agent can therefore make a different decision from the same evidence even though content integrity checks have nothing altered to detect. The attacker needs no access to the model, prompt, or tool-output contents; control over completion times, for instance through I/O or resource contention or the receiving infrastructure's completion queue, is sufficient.

For example, consider an agent choosing among three tree-planting programs based on four reports: equity, heat reduction, tree survival, and cost. The equity report favors one program, the heat report favors another, and the survival and cost reports favor the third. In our experiments, Qwen3 never selected the program favored by the equity report when the reports were presented in their natural order, with the equity report first. With the equity report last, the model selected that program in all 30 trials, despite receiving exactly the same report contents.

This journal version extends the original fixed-schedule attack in several ways. First, we formalize the delay-only threat model and distinguish *order sensitivity* (whether a decision depends on result order) from *steering* (whether an attacker can move it toward a chosen option). Second, we introduce an adaptive attacker that tests orders and selects one using observed decisions, exposing vulnerabilities a fixed order can miss. Third, we evaluate four defenses against both the attack and their own effects on clean decisions, including order-invariant synthesis, which summarizes each result independently before making the final decision. We also extend the empirical evaluation to additional models and decision tasks.

We evaluate ChronosAttack across six LLMs: GPT-5.6 Sol and GPT-5.6 Terra from OpenAI, Gemini 3.6 Flash from Google, Qwen3-235B from Alibaba, DeepSeek V4 Flash from DeepSeek, and Claude Sonnet 4.6 from Anthropic. Our evaluation includes six synthetic decision-making tasks: cloud backup, shipping partner selection, software release, micro-grid management, tree planting, and water supply. Each task requires the agent to choose among several options without a single objectively correct answer.

Our results show that all four models tested on tree planting were sensitive to the order of tool results. However, the

*Corresponding author: Arash Vashagh (arash.vashagh@unb.ca).

effectiveness of a particular attack order varied across models and depended on the decision context. The adaptive attacker also identified an effective ordering for a model that the fixed attacker failed to steer. Among the four defenses, only order-invariant synthesis consistently prevented substantial increases in the attacker's target selection.

Section II reviews related work. Sections III and IV define the attack and evaluation. Sections V–VII present the results, discuss their implications, and conclude.

II. Related Work

We relate ChronosAttack to content attacks on tool-using agents, order effects in language models, and timing attacks and defenses.

A. Tool-Using Agents and Content Attacks

Language models can learn when to call a tool and how to use its result [2], and agent frameworks interleave such calls with reasoning [1]. Recent benchmarks place agents in environments that change independently of the agent's actions [12], so the timing of observations becomes part of the task. Our setting shares this asynchrony, with the difference that an attacker controls the timing.

In the content attacks considered here, the adversary changes what the agent reads. Injected content can lead an agent into harmful actions [13], [14], and crafted tool descriptions and metadata can win tool selection or carry hidden instructions [15], [16]. Memory attacks insert records that later steer the agent [17], [18], and surveys organize these threats by where the attacker acts and what the attacker can access [19]–[21]. In contrast, the ChronosAttack adversary adds, alters, or removes nothing; the agent reads exactly the content that the tools produced.

B. Order and Position Effects

The sensitivity of language models to order is well documented. The order of in-context demonstrations can make the difference between strong and chance performance, and an order that works well for one model does not transfer to another [8]. Reordering answer options [9] or premises [22] changes accuracy, and relevant information is used less well in the middle of a long context than at its ends [10]. Similarly, LLM judges favor an answer because of its position [23], and the size of this positional bias varies across models [24].

These studies fix one prompt, permute its parts, and measure a loss of accuracy or consistency. Our work differs in three respects. First, the order is set by an attacker through tool completion times. Second, it unfolds over several agent turns, so earlier evidence can shape provisional choices before later evidence arrives. Third, the quantity of interest is how far the attacker can move the decision toward a chosen option.

C. Timing Attacks and Defenses

Timing has long been an attack surface in systems security. Time-of-check-to-time-of-use races, for example, exploit the interval between a check and the action it guards [25]. Reordering authentic events without modifying them has already been shown to defeat an adaptive monitor of electrocardiogram streams [26], and removing context while leaving the target untouched can change the label that a classifier assigns to it [27]. In agents, timing appears in asynchronous monitoring of actions [28] and in work showing that the moment at which an injected instruction appears affects its success [29]. In both cases, however, the attacker is the source of malicious behavior or content, whereas in ChronosAttack only the timing is adversarial.

Ren et al. study how asynchronous delivery changes the search queries and additional evidence acquired by search agents, and propose Confluent Evidence Acquisition to stabilize those paths [30]. Their focus is reproducible evidence acquisition and answer quality across schedules. We instead study an attacker who delays unchanged tool results to steer a decision toward a chosen option, even when the agent performs no further retrieval.

Several agent defenses address malicious instructions or their effects. They separate instructions from data [31], re-execute the agent to detect an injection's influence [32], check that each action serves the user's task [33], or attribute each tool call to its causes [34]. Like most defenses against adversarial inputs [35], each of them assumes a malicious input to find or isolate, and none constrains the order in which authentic results reach the model. The defenses we evaluate act on that order instead; one of them, voting over several orders, adapts the idea of aggregating answers over many samples [36].

D. Positioning of ChronosAttack

ChronosAttack connects order effects in LLMs to an adversary's control over an asynchronous agent's runtime. Unlike content attacks, it leaves every tool result authentic; unlike prompt-order studies, the adversary realizes an order through delays in tool completion. The closest work on asynchronous search agents examines how timing changes which evidence an agent subsequently acquires [30]. Our setting holds the available evidence fixed and asks whether its delivery order moves a final decision toward an attacker-selected target. This distinction motivates measuring both general sensitivity to order and targeted steering, and evaluating defenses by the decisions they produce as well as by whether they remove the attacker's control over order.

III. Methodology

We define the delay-only threat model, the fixed and adaptive attackers, and the metrics used to evaluate them. We then describe four defenses that change how the agent receives or combines tool results.

A. Threat Model

An agent must choose one option from a set $\mathcal{C}$. To do so, it requests n tool calls in parallel. The receiving infrastructure executes them, and tool i finishes with an authentic observation

o_i at time t_i. Sorting the completion times gives the natural order π_0, a permutation of $\{1, \ldots, n\}$. The runtime appends the observations to the model's context in a completion order π, so that $o_{\pi(k)}$ is the k-th observation the model reads; without an attacker, $\pi = \pi_0$. Fig. 1 illustrates this setting.

The attacker is any component that can slow down tool completion or reorder the completion queue on the receiving infrastructure. Examples include contention on local I/O or compute, a slow data source read by a tool, and the local task scheduler itself. The attacker adds a non-negative delay d_i to the completion of tool i, so that it finishes at

$$t'_i = t_i + d_i, \qquad 0 \le d_i \le D, \tag{1}$$

where D is the largest delay the runtime tolerates before treating a call as failed. The delivered order π is then the order of the t'_i in (1). The attacker cannot alter, add, remove, duplicate, or accelerate any observation. Hence, hashes and other integrity checks on the outputs pass. The attacker's goal is a target option $c^\star \in \mathcal{C}$, chosen in advance.

We consider two types of attackers: fixed and adaptive. The fixed attacker selects a tool-result order π_a in advance, based only on the available tools and decision options, and applies the same order across all models. In contrast, the adaptive attacker tests different orders, observes the agent's decisions, and selects the order $\pi^\star$ that most frequently leads to the target option $c^\star$. This attacker represents an adversary who can control tool-result scheduling and observe agent decisions over multiple interactions. Neither attacker can access the model weights or modify the prompt. We also assume agents do not receive tool completion timestamps, as these would provide timing information beyond the order of results.

B. Attack Design

We consider two types of agents: stateful and stateless. A stateful agent processes one observation per turn, updates its internal state, and records a provisional decision after each observation. The agent retains its previous context when processing subsequent observations:

$$h_k = f(h_{k-1} \oplus o_{\pi(k)}), \quad k = 1, \ldots, n; \quad y = g(h_n), \tag{2}$$

where h_0 is the task prompt, $\oplus$ denotes concatenation, f represents a model turn that updates the state and records a provisional decision, and g represents the final turn that produces the decision $y \in \mathcal{C}$.

In contrast, a stateless agent receives all observations in a single model call, arranged according to their delivery order:

$$y = g\big(h_0 \oplus o_{\pi(1)} \oplus \cdots \oplus o_{\pi(n)}\big). \tag{3}$$

In both settings, tool results are provided to the model as user messages, consistent with how tool results are returned to Anthropic models.[1] The agent is explicitly instructed not to interpret the arrival order of tool results as an indication of their importance, reliability, or relevance.

To manipulate the delivery order, the attacker introduces non-negative delays while preserving the original contents of all tool results. For a chosen ordering π, the attacker delays each tool completion until the preceding tool in that ordering has finished. The minimum delays required to achieve this ordering are calculated as

$$d_{\pi(k)} = \max\Big(0,\ t'_{\pi(k-1)} + \epsilon - t_{\pi(k)}\Big), \quad k = 2, \ldots, n, \tag{4}$$

with $d_{\pi(1)} = 0$, where ϵ is the smallest time gap required to distinguish consecutive completions. An ordering is feasible under the threat model only if every computed delay is at most D.

Because the agent does not receive completion timestamps, different feasible delay vectors that produce the same ordering result in identical model inputs. Consequently, ChronosAttack influences the agent's decisions through the order of tool results rather than their contents or the magnitude of the delays.

C. Evaluation Metrics

We evaluate ChronosAttack using metrics that capture both its ability to steer an agent toward a target option and the agent's general sensitivity to tool-result ordering.

To measure targeted steering, we evaluate each ordering π over R independent trials, producing decisions $y_\pi^{(1)}, \ldots, y_\pi^{(R)}$. The target-selection rate $\rho(\pi)$ and the effect of the fixed attack Δ are defined as

$$\rho(\pi) = \frac{1}{R} \sum_{r=1}^{R} \mathbf{1}\Big[y_\pi^{(r)} = c^\star\Big], \qquad \Delta = \rho(\pi_a) - \rho(\pi_0), \tag{5}$$

where $\mathbf{1}[\cdot]$ is the indicator function, π_a is the fixed attack order, and π_0 is the natural order. A positive observed value of Δ indicates an estimated increase in the probability of selecting the target option $c^\star$.

While the target-selection rate measures steering toward a particular option, it does not capture how sensitive the agent's decisions are to ordering in general. To evaluate order sensitivity, we consider an ordering Π drawn uniformly from all $n!$ possible permutations and the corresponding decision Y. We quantify the dependence between ordering and decision using mutual information:

$$I(\Pi; Y) = H(Y) - H(Y \mid \Pi), \tag{6}$$

where H denotes Shannon entropy in bits. A value of $I(\Pi; Y) = 0$ indicates that the decision is statistically independent of the ordering, whereas a positive value indicates that the ordering provides information about the agent's decision.

Because LLM responses can vary across repeated trials, we also distinguish changes caused by ordering from those arising through sampling variability. Specifically, we compare the probability of obtaining different decisions under the same ordering with the probability of obtaining different decisions under distinct orderings:

$$\begin{aligned} \delta_{\mathrm{w}} &= \Pr[Y \neq Y' \mid \Pi = \Pi'], \\ \delta_{\mathrm{b}} &= \Pr[Y \neq Y' \mid \Pi \neq \Pi'], \end{aligned} \tag{7}$$

where Y' denotes the decision of an independent trial under ordering Π'. When $\delta_{\mathrm{b}} > \delta_{\mathrm{w}}$, decisions differ more frequently

[1] https://platform.claude.com/docs/en/agents-and-tools/tool-use/handle-tool-calls

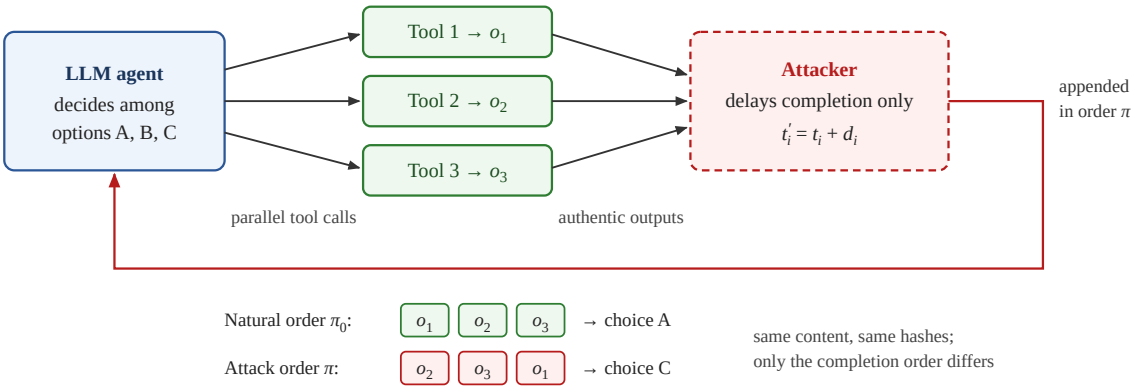


Fig. 1. The ChronosAttack threat model. The model requests parallel tool calls that run on the receiving infrastructure; the attacker delays tool completions without altering their outputs. The same authentic outputs, appended to the prompt in a different order, can lead to a different choice.

across distinct orderings than across repeated trials with the same ordering, providing additional evidence of order sensitivity.

Finally, we examine how the extent of reordering relates to attack effectiveness. We measure this using the inversion distance $\kappa(\pi, \pi_0)$, defined as the number of pairs of observations whose relative order differs between the delivered order π and the natural order π_0. This metric allows us to investigate whether larger deviations from the natural ordering are associated with greater changes in the agent's decisions.

D. Order-Based Defenses

The four defenses we evaluate all act on the order in which the model reads the results. The barrier waits for all n results and presents them together in a fixed canonical order π_c. Randomized ordering presents the results in a fresh random order for every decision, so that chance replaces the attacker's order. The order vote presents the results in several orders, each as a single joint prompt, and returns the majority choice if it reaches a quorum and abstains otherwise. The fourth defense, order-invariant synthesis, removes the order from the decision input altogether. Each observation is first summarized on its own, in a separate call that sees no other observation, giving $s(o_i)$. The decision is then made from the summaries in the canonical order:

$$y = g\big(h_0 \oplus s(o_{\pi_c(1)}) \oplus \cdots \oplus s(o_{\pi_c(n)})\big)\,. \tag{8}$$

Since each summary is computed independently and the summaries are presented in a fixed order, the input of the final call does not depend on π, as shown in Fig. 2.

As a result, none of the four defenses lets the attacker choose the order that the model reads: the barrier and the vote ignore π, randomized ordering replaces it, and (8) removes it. The defenses differ, however, in what the model reads instead, and that input can itself move the decision. We therefore judge each defense by whether the target rate it produces stays at the natural rate $\rho(\pi_0)$ of (5).

IV. Experimental Setup

We use six synthetic decision tasks, each with three options, and fix the target $c^\star$ before any model call. The cloud-backup and shipping-partner tasks come from the preprint [11] and have three tools. The software-release and microgrid tasks also have three tools and pose a comparable trade-off in new domains. The tree-planting task has four tools, which report equity, heat reduction, tree survival, and cost, and each option is best on at least one of them. The water-supply task repeats this four-tool structure with aquifer protection, drought yield, lifecycle cost, and time to supply. In the four-tool tasks, the fixed order π_a moves the report that favors the target from first to last. For each prespecified experiment, the payloads, prompts, and candidate orders were frozen and hashed before its collection; the adaptive order was selected from a completed 24-order panel and tested on fresh trials. Since no option is correct in any task, the target rate (5) measures how far the decision can be moved, not how often it is wrong.

The original study ran GPT-5.6 Sol, Gemini 3.6 Flash, DeepSeek V4 Flash, and Claude Sonnet 4.6, and we take its numbers from the preprint. The new experiments ran on Kaggle Benchmarks with GPT-5.6 Terra, Gemini 3.6 Flash, Qwen3-235B-A22B-Instruct-2507, and Claude Sonnet 4.6. Because the platform hosts neither GPT-5.6 Sol nor DeepSeek V4 Flash, the new evaluation uses Terra and Qwen rather than those models. We used the lowest accepted non-off reasoning settings for Terra and Qwen (low) and Gemini (minimal). Claude ran with reasoning off. Its minimal setting failed on every request, and with its low setting one trial spent the whole answer cap on reasoning in all three attempts, so that cell could not be completed. Every call used temperature 1 (as in the preprint), an answer cap of 256 tokens (enough for a choice and a short reason), a timeout of 30 seconds, and no seed. Every trial began a new chat. We replayed each schedule by presenting the same authentic payloads to the model in the specified order. Arrival times and added delays were computed and logged, rather than imposed on live tool executions; the models did not receive those timestamps. The experiments therefore isolate the effect of delivered order, rather than measuring end-to-end exploitation of a deployed tool runtime.

For each model and task, we ran $R = 30$ independent

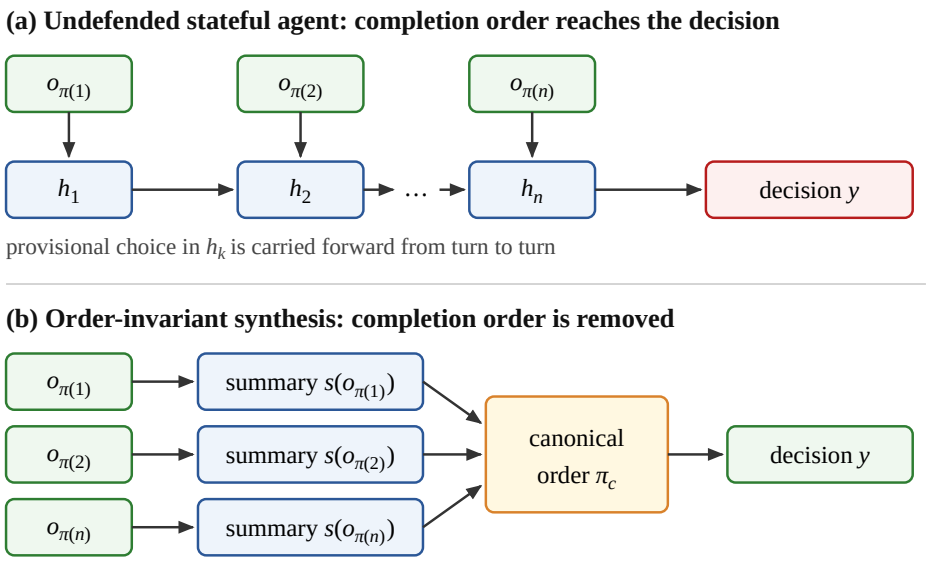


Fig. 2. Comparison of an undefended stateful agent and order-invariant synthesis. (a) The undefended agent processes tool results as they arrive, updating its state and provisional decision after each result. Consequently, changing the arrival order can affect its final decision. (b) In order-invariant synthesis, each tool result is summarized independently, and the summaries are presented in a fixed order before the final decision is made. This prevents the tool completion order from determining the order of information presented for the final decision.

stateful trials under each of π_0 and π_a. Each condition used new chats and model calls; matching repetition numbers were job identifiers, not a pairing mechanism. We also ran ten stateless trials per order, ten trials with all observations presented at once and no completion order, 15 stateful trials at each inversion distance, and 15 barrier trials. In the four-tool tasks, we additionally ran all 24 orders ten times each, both statefully and as one joint prompt. These 24-order panels give $I(\Pi; Y)$ and the adaptive attacker's order $\pi^\star$, which was then tested against each defense on 30 fresh trials per condition. The barrier used the natural order as its canonical order, and the order vote used all 24 orders with a quorum of 16 votes. Natural completion times in the four-tool tasks were 50, 100, 150, and 200 ms, and ϵ was 1 ms. Fixed-order effects were tested with two-sided Fisher exact tests on the independent target-choice counts, with Holm correction [37] across the 16 new model-task comparisons in Table I. We use 95% Newcombe score intervals for differences between independent proportions [38]. The original preprint cells are descriptive here. $I(\Pi; Y)$ was tested against 999 shuffles of the orders within each panel, and differences between models in the effect of the order were tested with Firth-penalized logistic regression [39]. The stateful comparison of π_0 and π_a is the confirmatory endpoint; the adaptive attacker's order and every analysis chosen after seeing the data are exploratory. For the new experiments, every call was logged before analysis and every number reported below is recomputed from those logs. The ablation analysis in Section V-A examines sensitivity to these settings.

V. Results

We first evaluate the fixed attack's effectiveness. Table I compares the target-selection rates under the natural and fixed attack orders, using 30 independent trials per order for each model-task combination. Part (a) presents the results from our original study [11] descriptively, while part (b) reports the new experiments. In part (b), bold marks differences significant under two-sided Fisher exact tests with Holm correction across the 16 cells ($p < 0.05$). A fixed order steered at least one model in every new task, but no fixed order steered every model. Qwen was steered in all four new tasks, from 0% to 100% in three of them. Claude moved by 70 and 53 points in software release and microgrid, Terra by 37 points in software release, and Gemini by 47 points in tree planting. Where a model already chose the target, as Terra, Gemini, and Claude did in water supply, the attack had no room to act. The same order can also push a model away from the target, as it pushed Gemini from 100% to 7% in cloud backup. Sequential state is not required either. In the stateless setting (3), Claude's cloud-backup choice moved from 0 to 10 of 10 trials, and in tree planting Qwen was steered by one joint prompt. For Qwen, the effect is also not tied to the wording: its steering was unchanged under two rewordings of the prompts and reports, and under two relabelings of the options it moved from 0% to 100% in one and from 53% to 77% in the other. Part of Gemini's effect, in contrast, followed the option letter, falling to 3% under one relabeling.

Because all 24 orders were run in tree planting, this task also lets us separate order sensitivity from steering. Table II shows that every model's choice depends on the order. $I(\Pi; Y)$ (6) ranges from 0.15 to 0.90 bits, each value with permutation $p = 0.001$, and for every model two decisions under different orders disagree more often than two decisions under the same order (7). Fig. 3 gives the target rate for each of the 24 orders, and the last column of Table II summarizes it: for every model, the target rate ranges from 0% under some orders to 40% for

TABLE I
FIXED ATTACK: STATEFUL TARGET RATE (%), NATURAL → FIXED ORDER

(a) Original tasks and models

Model	Cloud backup	Shipping partner
GPT-5.6 Sol	0→83	10→67
Gemini 3.6	100→7	100→80
DeepSeek V4	7→13	13→20
Claude 4.6	47→93	0→0

(b) New tasks and models

Model	Release	Microgrid	Tree	Water
GPT-5.6 Terra	**57→93**	80→90	3→3	100→100
Gemini 3.6	93→90	10→30	**47→93**	100→100
Qwen3	**0→100**	**0→70**	**0→100**	**0→100**
Claude 4.6	**20→90**	**13→67**	93→77	100→100

Terra and 100% for the other models under others. Steering toward the target, however, needs room to move. Terra chooses the target in at most 40% of trials under any order, and Claude already chooses it under the natural order. The effect of the order also differs between models (penalized likelihood-ratio $\chi^2(69) = 327.9$, asymptotic $p = 2.4 \times 10^{-35}$; exploratory, with uncertain calibration in this sparse panel), and so does the observation that carries it. For Qwen, moving the equity report from first to last raises the target rate by 100 points, whereas Gemini and Claude respond mostly to the positions of the heat and cost reports.

TABLE II
TREE PLANTING: ORDER SENSITIVITY AGAINST SAMPLING NOISE OVER ALL 24 ORDERS

Model	$I(\Pi; Y)$ (bits)	δ_{w}	δ_{b}	Range of ρ (%)
GPT-5.6 Terra	0.15	0.10	0.13	0–40
Gemini 3.6	0.49	0.09	0.30	0–100
Qwen3	0.90	0.07	0.52	0–100
Claude 4.6	0.40	0.27	0.47	0–100

Since the fixed order is chosen without seeing any outcome, we next turn to the adaptive attacker. Fig. 4 shows its results together with those of the defenses, with 30 trials per condition. The adaptive attacker steered Terra, which the fixed order had not moved, and moved Gemini and Qwen about as far as the fixed order had. On fresh trials, its order $\pi^\star$ raised the target rate of Terra from 0% to 60%, of Gemini from 37% to 100%, and of Qwen from 0% to 97%. For Claude, which already chose the target under the natural order, the search found no stronger order. Because no defense lets the attacker choose the order, each defense gives the same input distribution against both attackers, and the defenses differ only in where they leave the decision. Order-invariant synthesis (8) removed the fixed attack's effect on both steered models, from 27 to 0 of 30 trials for Gemini and from 30 to 0 for Qwen, and against $\pi^\star$ it kept every model at or below its natural rate, apart from 3% against 0% for Terra. Each of the other defenses, on the other hand, led at least one model to the target by itself. The barrier's single prompt in the natural order did so for Gemini in 60% of trials, against a natural rate of 37%, and in 8 of 15 trials of the main study. Randomized ordering did so for Qwen in 7 of 30 trials under the fixed attack and in 40% here, since about half of Qwen's 24 orders lead to the target, as Fig. 3 shows. The vote chose the target for Qwen in 9 of 10 order panels and in 70% of decisions here, abstaining on the rest. Finally, order-invariant synthesis used five planned model calls per decision for Terra and Claude and six for Gemini and Qwen, compared with five for the undefended agent and 24 for the vote. Synthesis changed Gemini's clean choice distribution by 0.30 in total variation.

A. Ablations and Hyperparameter Sensitivity

We examine sensitivity to attack, model, and sampling settings.

1) Attack Parameters: Because models see neither timestamps nor delays, feasible delay vectors producing the same order give identical inputs. Equation (4) gives the minimum delays. Simulated completion times of 50, 100, 150, and 200 ms and of 2,350, 4,200, 6,050, and 7,900 ms produced byte-identical prompts under one order. For the fixed attack schedules, maximum computed delays were 136 and 66 ms in the original tasks and 151 ms in the four-tool tasks. These simulations do not establish a deployed runtime's tolerance D.

Table III reports tree-planting target rates by inversion distance $\kappa(\pi, \pi_0)$, using 15 stateful trials of the lexicographically first order at each distance, chosen without outcomes. The fixed attack has distance 3. Qwen moves from 0% at distances 0–3 to 100% at 4–6; other models are nonmonotonic. Claude, for example, falls to 0% at distance 3 and returns to 93% at 6. The preprint likewise found that one inversion could be as effective as two [11]. Thus, distance alone does not characterize an attack order.

TABLE III
TREE PLANTING: TARGET RATE (%) BY INVERSION DISTANCE $\kappa(\pi, \pi_0)$

Model	0	1	2	3	4	5	6
GPT-5.6 Terra	7	0	0	7	0	0	0
Gemini 3.6	40	0	40	0	0	0	0
Qwen3	0	0	0	0	100	100	100
Claude 4.6	87	93	67	0	67	80	93

Removing each tree-planting report in turn leaves three observations and six orders. We tested every order ten times and averaged across the four removals, retaining the other payloads unchanged (Table IV). Excess order information is $I(\Pi; Y)$ minus its shuffled-order mean. With three reports, it falls near zero for Terra and Gemini; Gemini's highest target rate falls from 100% to 32%. Qwen remains more sensitive, while Claude shows a smaller decline. Removing evidence also changes the decision trade-off, and the four-report runs preceded the three-report runs by a month, so this comparison does not isolate observation count.

2) Model and Sampling Parameters: Table V compares Terra with reasoning off and low (the main setting), using 30 trials per order and the 24-order panels. The attack increased target rates by 83 and 53 points in release and microgrid with reasoning off, versus 37 and 10 with low reasoning, partly because the low-reasoning natural rates were higher. Tree-planting target rates remained near zero, and model-by-order

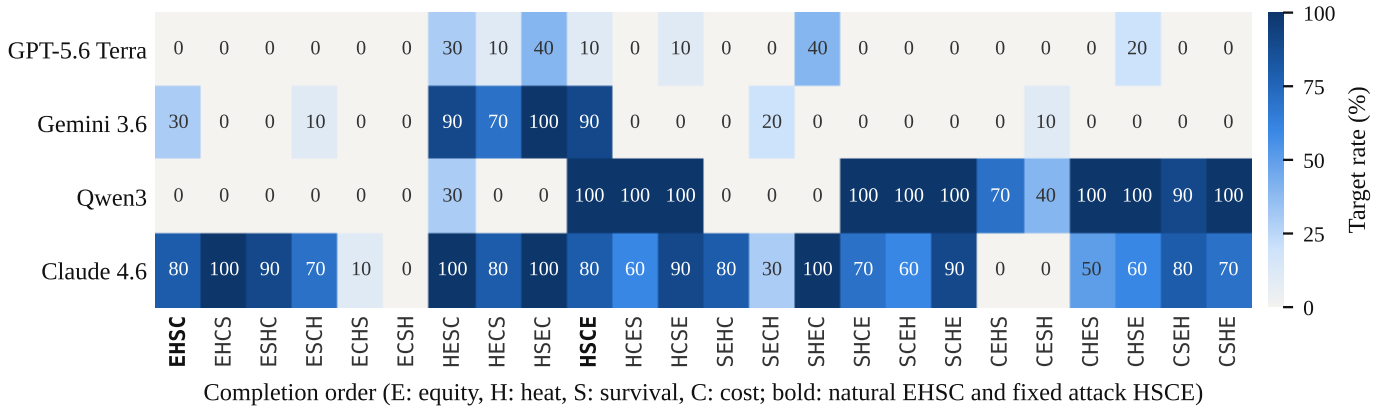


Fig. 3. Target-selection rates in the tree-planting task across all 24 possible tool-result orders, with ten stateful trials per order. Each ordering is represented by the sequence of four reports: equity (E), heat (H), survival (S), and cost (C). The natural order is EHSC, while the fixed attack order is HSCE.

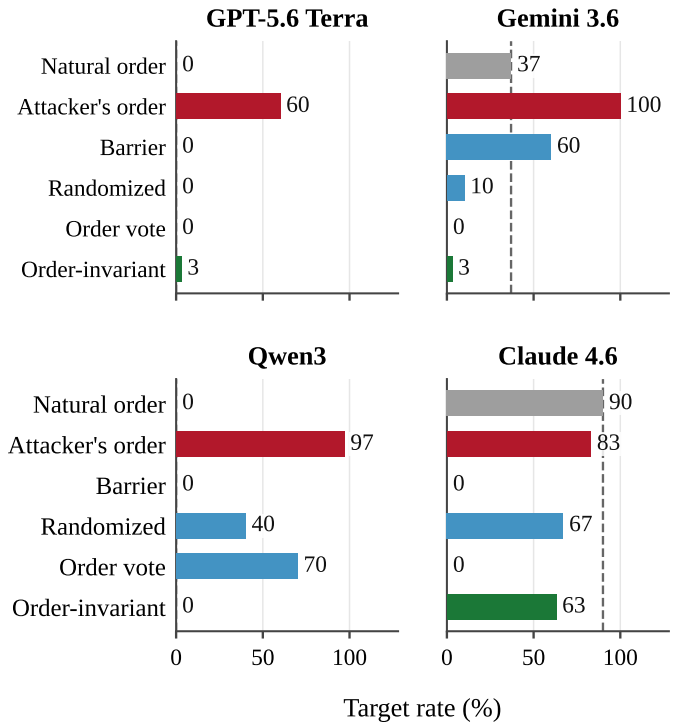


Fig. 4. Tree planting: target rate (%) under each model's natural order, the adaptive attacker's order $\pi^\star$, and $\pi^\star$ with each defense applied on its own. The dashed line marks the natural rate. $\pi^\star$ was chosen after seeing each model's 24-order panel (exploratory).

TABLE IV
TREE PLANTING: FOUR AGAINST THREE OBSERVATIONS

Model	Excess $I(\Pi;Y)$ (bits) Four	Excess $I(\Pi;Y)$ (bits) Three	Highest ρ (%) Four	Highest ρ (%) Three
GPT-5.6 Terra	0.07	0.00	40	25
Gemini 3.6	0.40	0.02	100	32
Qwen3	0.80	0.48	100	75
Claude 4.6	0.32	0.20	100	82

statistics were similar ($\chi^2(69) = 319.7$ versus 327.9). The runs were a month apart, so the comparison is descriptive.

With $R = 30$ independent trials per order, every Holm-significant effect among the 16 new comparisons was at least 37 points. Gemini's 20-point microgrid increase was not significant (95% independent-proportions interval: -1 to $+39$ points). An order selected from five of ten trials per permutation raised the target rate on the remaining five by 100 points for Qwen and 60 for Gemini. This split-half check is descriptive; Fig. 4 reports 30 fresh trials per condition. The 256-token answer cap also constrained Claude's reasoning settings (Section IV).

TABLE V
GPT-5.6 TERRA: REASONING OFF AGAINST LOW

Setting	Release (%)	Microgrid (%)	Tree (%)	Tree $I(\Pi;Y)$
Off	17→100	33→87	0→7	0.09
Low	57→93	80→90	3→3	0.15

VI. DISCUSSION AND FUTURE DIRECTIONS

These results support reporting order sensitivity and steering as separate properties. Every model in tree planting changed its decision with the order of the same evidence, yet a given order steered only some models in some tasks, and the direction of the shift differed between models. A single attack success rate, measured with one order on one model, would describe neither property. An evaluation is therefore more informative when it reports, for each model and task, the order information together with the effect of a fixed order and of the order found by search.

Steering also depends on the model's natural choice. In the water-supply task, three of the four models already selected the target under the natural order, leaving no room to increase its selection rate. The ablations in Section V-A also show that removing one report substantially reduced order sensitivity for Terra and Gemini in tree planting.

Most models changed their provisional decisions approximately once per trial, while Qwen did so approximately twice. These changes occurred most frequently after the second observation, and the final decision rarely differed from the last provisional choice. These trajectories suggest that evidence received at some stages may influence the final choice more

than evidence received at others. Gemini's sensitivity to option labels also suggests that ordering effects may depend on how the choices are presented.

The study uses synthetic decisions without an objectively correct option and replays tool-result orders rather than delaying live tools. It establishes decision sensitivity under controlled evidence but does not measure task accuracy or whether an attacker can induce the required delays in a deployed runtime.

Experiments with agents that use external services or execute complex workflows could test whether the observed effects persist under realistic network delays and resource constraints.

Adaptive attacks on previously unseen agents also need study. Measuring how many observations of agent decisions an attacker needs to find an effective order would help assess attacks with limited feedback.

An adaptive defense could summarize each tool result independently and request further analysis only when the evidence conflicts or the decision remains uncertain. Whether this reduces cost while preserving resistance to scheduling attacks should be tested together with latency and clean decision quality.

VII. Conclusion

ChronosAttack shows that an agent's final choice can depend on the delivery order of authentic tool results. The delay-only threat model separates this scheduling attack from content manipulation and distinguishes general order sensitivity from steering toward a chosen option. In the evaluated tasks, a model that resisted one fixed order could still be steered by an order found through search. Removing the attacker's control over delivery order also did not guarantee preservation of the agent's clean decisions. Order-invariant synthesis most consistently prevented targeted steering in the tree-planting experiments, but changed some models' choices without an attack. These controlled replay results motivate evaluation in deployed agents, where the attacker's ability to induce the necessary delays remains to be established.

References

[1] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. R. Narasimhan, and Y. Cao, "ReAct: Synergizing reasoning and acting in language models," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2023.

[2] T. Schick, J. Dwivedi-Yu, R. Dessì, R. Raileanu, M. Lomeli, E. Hambro, L. Zettlemoyer, N. Cancedda, and T. Scialom, "Toolformer: Language models can teach themselves to use tools," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2023.

[3] S. Kim, S. Moon, R. Tabrizi, N. Lee, M. W. Mahoney, K. Keutzer, and A. Gholami, "An LLM compiler for parallel function calling," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2024, pp. 24 370–24 391.

[4] I. Gim, S. Lee, and L. Zhong, "Asynchronous LLM function calling," arXiv:2412.07017, 2024.

[5] K. Greshake, S. Abdelnabi, S. Mishra, C. Endres, T. Holz, and M. Fritz, "Not what you've signed up for: Compromising real-world LLM-integrated applications with indirect prompt injection," in *Proc. 16th ACM Workshop Artif. Intell. Secur. (AISec)*, 2023, pp. 79–90.

[6] E. Debenedetti, J. Zhang, M. Balunovic, L. Beurer-Kellner, M. Fischer, and F. Tramèr, "AgentDojo: A dynamic environment to evaluate prompt injection attacks and defenses for LLM agents," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS Datasets Benchmarks Track)*, 2024.

[7] E. Debenedetti, I. Shumailov, T. Fan, J. Hayes, N. Carlini, D. Fabian, C. Kern, C. Shi, A. Terzis, and F. Tramèr, "Defeating prompt injections by design," in *Proc. IEEE Conf. Secure Trustworthy Mach. Learn. (SaTML)*, 2026, pp. 587–618.

[8] Y. Lu, M. Bartolo, A. Moore, S. Riedel, and P. Stenetorp, "Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity," in *Proc. 60th Annu. Meeting Assoc. Comput. Linguistics (ACL)*, 2022, pp. 8086–8098.

[9] P. Pezeshkpour and E. Hruschka, "Large language models sensitivity to the order of options in multiple-choice questions," in *Findings Assoc. Comput. Linguistics (NAACL)*, 2024, pp. 2006–2017.

[10] N. F. Liu, K. Lin, J. Hewitt, A. Paranjape, M. Bevilacqua, F. Petroni, and P. Liang, "Lost in the middle: How language models use long contexts," *Trans. Assoc. Comput. Linguistics*, vol. 12, pp. 157–173, 2024.

[11] A. Vashagh, "ChronosAttack: Adversarial tool scheduling attacks on LLM agents," Research Square preprint, 2026.

[12] R. Froger, P. Andrews, M. Bettini, A. Budhiraja, R. S. Cabral, V. Do, E. Garreau, J. Gaya, H. Laurençon, M. Lecanu, K. Malkan, D. Mekala, P. Ménard, G. M. Bertran, U. Piterbarg, M. Plekhanov, M. Rita, A. Rusakov, V. Vorotilov, M. Wang, I. Yu, A. Benhalloum, G. Mialon, and T. Scialom, "Gaia2: Benchmarking LLM agents on dynamic and asynchronous environments," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2026.

[13] Q. Zhan, Z. Liang, Z. Ying, and D. Kang, "InjecAgent: Benchmarking indirect prompt injections in tool-integrated large language model agents," in *Findings Assoc. Comput. Linguistics (ACL)*, 2024, pp. 10 471–10 506.

[14] I. Nakash, G. Kour, G. Uziel, and A. Anaby-Tavor, "Breaking ReAct agents: Foot-in-the-door attack will get you in," in *Findings Assoc. Comput. Linguistics (NAACL)*, 2025, pp. 6484–6509.

[15] J. Shi, Z. Yuan, G. Tie, P. Zhou, N. Z. Gong, and L. Sun, "Prompt injection attack to tool selection in LLM agents," in *Proc. Netw. Distrib. Syst. Secur. Symp. (NDSS)*, 2026.

[16] Z. Wang, Y. Gao, Y. Wang, S. Liu, H. Sun, H. Cheng, G. Shi, H. Du, and X. Li, "MCPTox: A benchmark for tool poisoning on real-world MCP servers," in *Proc. AAAI Conf. Artif. Intell.*, 2026, pp. 35 811–35 819.

[17] S. Dong, S. Xu, P. He, Y. Li, J. Tang, T. Liu, H. Liu, and Z. Xiang, "Memory injection attacks on LLM agents via query-only interaction," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2025.

[18] Z. Chen, Z. Xiang, C. Xiao, D. Song, and B. Li, "AgentPoison: Red-teaming LLM agents via poisoning memory or knowledge bases," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2024.

[19] M. A. Ferrag, N. Tihanyi, D. Hamouda, L. Maglaras, A. Lakas, and M. Debbah, "From prompt injections to protocol exploits: Threats in LLM-powered AI agents workflows," *ICT Express*, vol. 12, no. 2, pp. 353–383, 2026.

[20] J. Kim, W. Guo, and D. Song, "SoK: Attack and defense landscape of agentic AI systems," in *Proc. 35th USENIX Secur. Symp.*, 2026, pp. 6047–6066.

[21] A. Vashagh, R. Razavi-Far, M. Meymani, and B. Biggio, "Recent advances in adversarial attacks on model utility, privacy, and explainability: A comprehensive survey," TechRxiv preprint, 2026.

[22] X. Chen, R. A. Chi, X. Wang, and D. Zhou, "Premise order matters in reasoning with large language models," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2024, pp. 6596–6620.

[23] Z. Li, C. Wang, P. Ma, D. Wu, S. Wang, C. Gao, and Y. Liu, "Split and merge: Aligning position biases in LLM-based evaluators," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, 2024, pp. 11 084–11 108.

[24] P. Schilcher, D. Karasin, M. Schöpf, H. Saleh, A. Tommasel, and M. Schedl, "Characterizing positional bias in large language models: A multi-model evaluation of prompt order effects," in *Findings Assoc. Comput. Linguistics (EMNLP)*, 2025, pp. 20 643–20 664.

[25] D. Dean and A. J. Hu, "Fixing races for fun and profit: How to use access(2)," in *Proc. 13th USENIX Secur. Symp.*, 2004, pp. 195–206.

[26] A. Vashagh and Y. Vashagh, "ConformalShift: Targeted event reordering against adaptive ECG monitoring," arXiv:2608.03628, 2026.

[27] ——, "CohortHijack: Robustness of single cell annotation to companion cell removal," arXiv:2608.05900, 2026.

[28] A. C. Stickland, J. Michelfeit, A. Mani, C. Griffin, O. Matthews, T. Korbak, R. Inglis, O. Makins, and A. Cooney, "Async control: Stress-testing asynchronous control measures for LLM agents," arXiv:2512.13526, 2025.

[29] Y. Mou, P. Yang, Z. Yin, Z. Xue, X. Luan, D. Yu, T. Zhang, S. Zhang, and W. Ye, "ToolHazard: Scaling adversarial environments for security evaluation and alignment of LLM-based agents," arXiv:2608.11878, 2026.

[30] R. Ren, Y. Wang, Y. Zhang, and C. Miao, "The scheduler is a hidden retriever: How asynchrony shapes evidence acquisition in search agents," *arXiv preprint arXiv:2610.26310*, 2026.

[31] S. Chen, J. Piet, C. Sitawarin, and D. A. Wagner, "StruQ: Defending against prompt injection with structured queries," in *Proc. 34th USENIX Secur. Symp.*, 2025, pp. 2383–2400.

[32] K. Zhu, X. Yang, J. Wang, W. Guo, and W. Y. Wang, "MELON: provable defense against indirect prompt injection attacks in AI agents," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2025.

[33] F. Jia, T. Wu, X. Qin, and A. C. Squicciarini, "The task shield: Enforcing task alignment to defend against indirect prompt injection in LLM agents," in *Proc. 63rd Annu. Meeting Assoc. Comput. Linguistics (ACL)*, 2025, pp. 29 680–29 697.

[34] Y. He, H. Zhu, Y. Li, S. Shao, H. Yao, Z. Liu, and Z. Qin, "AttriGuard: Defeating indirect prompt injection in LLM agents via causal attribution of tool invocations," in *Proc. 35th USENIX Secur. Symp.*, 2026, pp. 1547–1566.

[35] M. Meymani, R. Razavi-Far, A. Vashagh, and B. Biggio, "Defense against adversarial attacks: Foundations, strategies, and future directions," Preprints.org preprint, 2026.

[36] X. Wang, J. Wei, D. Schuurmans, Q. V. Le, E. H. Chi, S. Narang, A. Chowdhery, and D. Zhou, "Self-consistency improves chain of thought reasoning in language models," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2023.

[37] S. Holm, "A simple sequentially rejective multiple test procedure," *Scandinavian Journal of Statistics*, vol. 6, no. 2, pp. 65–70, 1979.

[38] R. Newcombe, "Interval estimation for the difference between independent proportions: Comparison of eleven methods," *Statistics in medicine*, vol. 17, pp. 873–90, 1998.

[39] D. Firth, "Bias reduction of maximum likelihood estimates," *Biometrika*, vol. 80, no. 1, pp. 27–38, 1993.