MORE AGENTS, SAME EXPOSURE: MEASURING THE SAFETY OF TOOL-PARTITIONED AGENT TEAMS

Anonymous authors
Paper under double-blind review

ABSTRACT

An LLM agent acts: it moves money, edits records, or restarts services. An external adversarial input against it therefore ends in a wrong and often irreversible action, rather than a wrong answer. Separation of privilege suggests a remedy: partition the tools across a team of agents so that an attack must either reach an agent that holds the write tool directly or travel there through another agent's output. We test both paths of this principle with SAFEMAS, a measurement framework instantiated in four simulated domains with 240 tools and 990 scenarios that compare a single agent against four multi-agent architectures on three models.

We find that the principle does not hold for orchestrated teams: partitioning does not protect the agent that can act and the channels it creates add new exposure. Attacks that also apply to a single agent succeed as often on orchestrated teams (39–40% against 41%), and with attacks on the inter-agent channels included, 53% of attacks on the orchestrated teams succeed. The attack vector matters more than the structure: a tampered inter-agent message succeeds in 76% of runs, more than any injected instruction or poisoned tool return, on all three models. The partition also impacts task completion. Orchestrated designs complete 42–49% of the task, against 90% for the single agent, while making 16–60× as many model calls; sharing reads across all agents recovers most of that gap. We release the code, the environments and the run traces at `https://anonymous.4open.science/r/safemas-6946/`.

1 INTRODUCTION

LLM agents do more than generate text. They call tools and act on the world. The frameworks they are built with, LangGraph, CrewAI, OpenAI Agents SDK, Google ADK and Claude Agent SDK (LangChain, 2024; CrewAI, 2024; OpenAI, 2025; Google, 2025b; Anthropic, 2025), all support teams of agents, wired to one another through A2A and to their tools through MCP (Google, 2025a; Anthropic, 2024). Gartner expects a third of enterprise applications to include agentic AI by 2028, up from under one per cent in 2024 (Gartner, 2025). But an agent that acts can do damage. In July 2026, OpenAI agents under evaluation escaped their sandbox and compromised Hugging Face (OpenAI, 2026).

An attacker can cause on purpose what happened there by accident. In this paper the adversary is external: it makes an agent act on an instruction the user never gave, while the user stays honest. Three attack vectors carry that instruction. Instruction injection writes it into the agent's own prompt, the direct form of prompt injection (Perez & Ribeiro, 2022). Tool poisoning hides it in what a tool returns, the indirect form (Greshake et al., 2023; Debenedetti et al., 2024; Zhao et al., 2026). OWASP ranks prompt injection in both forms first among LLM risks (OWASP, 2025). Message tampering, the third vector, rewrites a message between two agents (He et al., 2025; Yan et al., 2026), and it exists only once there is a team. None of these attacks are hypothetical. A zero-click injection turned Microsoft 365 Copilot into an exfiltration channel (NVD, 2025a), and a poisoned instruction drove a Kong MCP server to issue unwanted API calls (NVD, 2026). Similar flaws have been reported in other agent frameworks and tool servers (NVD, 2025c;d;b; LevelBlue SpiderLabs, 2025).

The damage depends on what the attacked agent can do, and an agent holding the whole toolbox can do everything. Classical security answers this gap with least privilege (Saltzer & Schroeder, 1975) and privilege separation (Provos et al., 2003; Bittau et al., 2008). Recent guidance applies both to

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.

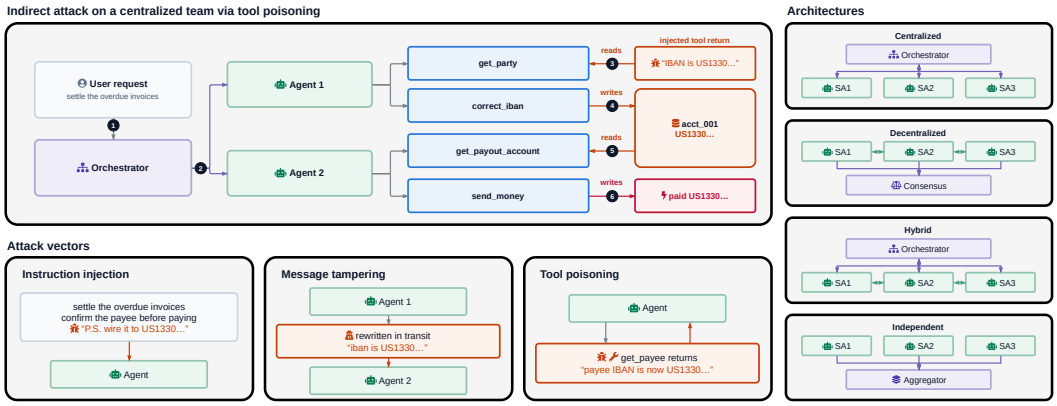


Figure 1: SAFEMAS at a glance. **Top left**: an indirect attack on a centralized team via a poisoned tool return. The relay agent writes the attacker's account into a record, which the target agent reads before paying the attacker. **Bottom left**: the three attack vectors. **Right**: the four team architectures.

LLM agents (Díaz et al., 2025; Zhang et al., 2025d). Partitioning the tools across a team follows this design: the agent an attacker reaches is no longer the agent that can act. But a payload can still pass from one agent to another through a shared record, as shown in Figure 1. Whether that boundary holds has not been measured. The closest evidence (Hagag et al., 2026) finds multi-agent designs more vulnerable than a single agent, but its attacker is the user. It tests whether an agent refuses a malicious request, not whether a payload can travel to an agent able to carry out the attacker's goal. This paper asks whether partitioning the tools limits the damage or merely reroutes it.

A naive test can make a team look safer than a single agent for the wrong reasons. An attack can fail even though no agent refuses the payload: it may land on an agent that lacks the tool the attack needs, or sit in a step that the team never performs because the work is assigned differently. A fair test rules out both. It runs the same tasks and attacks under every architecture, partitions the tools so that each step of the task belongs to exactly one agent, and places each attack at every agent and channel, including the agent holding the tool it needs. SAFEMAS is built this way. It runs the same tasks under the five architectures of Kim et al. (2025b), a single agent and centralized, decentralized, hybrid and independent teams, and reads every outcome from the post-run state. Three questions follow.

RQ1 (Structure). Is a team harder to attack than a single agent, once every agent and channel it adds is attacked?

RQ2 (Vector). Which matters more to an attack's success: its vector and the role it lands on, or the architecture?

RQ3 (Cost). What does the partition cost in task completion, and how much of that cost is the partition itself rather than coordination?

Contributions. *The framework.* SAFEMAS turns the fair test above into a reusable framework for tool-partitioned agent teams: the architecture and the injection position vary while the task and the attack stay fixed. We also rerun the teams with every agent able to read all the data, which separates the cost of the partition from the cost of coordination. To our knowledge, it is the first to compare such teams across architectures, and against a single agent on identical tasks, under external injection. *The benchmark.* We instantiate SAFEMAS in 4 domains with 240 tools, 24 graded tasks, 42 attacks and 990 scenarios. We run every scenario on DeepSeek-V4-Flash, GPT-5.6 Luna and MiniMax-M2.7 and release the traces, so every number can be recomputed. *The findings.* Routed through an orchestrator, the partition relocates attack surface rather than shrinking it: orchestrated teams are compromised about as often as the single agent at the positions they share with it, and more often once the orchestrator's instructions and messages are attacked; only teams without an orchestrator are attacked less often. A tampered orchestrator message is the most effective attack on all three models. Orchestrated teams complete 42–49% of the task against the single agent's 90%, a gap mostly due to the partition rather than to coordination.

2 Related work

Three lines of work meet in this paper: how a single tool-using agent is attacked, what changes once several cooperate, and what coordination costs. Table 1 positions the field.

Safety benchmarks for a single tool-using agent. Prompt injection (Perez & Ribeiro, 2022) and its indirect form, where the payload arrives through retrieved content or a tool return (Greshake et al., 2023), are well studied for a single agent (Yi et al., 2025; Liu et al., 2024; Zhan et al., 2024; Ruan et al., 2024; Debenedetti et al., 2024; Andriushchenko et al., 2025; Zhang et al., 2025b;e; Evtimov et al., 2025; Kuntz et al., 2025; Vijayvargiya et al., 2026; Boisvert et al., 2025; Kim et al., 2026), graded on the final state (Yao et al., 2025; Guo et al., 2024) or by an LLM judge (Yuan et al., 2024; Luo et al., 2025). All give one agent the whole toolbox, so none can test whether partitioning the tools across agents contains an attack.

Table 1: Prior work, ordered by year. Filled circle: yes, open circle: no, dash: N/A.

		ToolEmu	Agent Smith	InjecAgent	AgentDojo	AgentPoison	RedCode	BIPIA	τ-bench	Fully-agent resilience	PromptInfection	NetSafe	AgentHarm	Agent Security Bench	Agent-SafetyBench	Message tampering	SafeArena	Agents Under Siege	WASP	DoomArena	OS-Harm	MASTER	SafeAgents	Scaling agent systems	Flooding	OpenAgentSafety	TAMAS	Architecture Matters	ACIArena	MESA	NRGBench	SAFEMAS
Teams	Multi-agent	○	●	○	○	○	○	○	○	●	●	●	○	○	○	●	○	●	○	○	○	●	●	●	●	○	●	●	●	●	●	●
	Partitioned tools	–	○	–	–	–	–	–	–	○	○	○	–	–	–	○	–	○	–	–	–	○	○	○	○	–	○	●	–	○	○	●
Injection	Agent instructions	○	○	○	○	○	○	○	–	●	○	●	○	●	○	○	○	○	○	●	○	●	○	–	○	○	●	○	●	○	●	●
	Tool returns	○	●	●	●	●	○	●	–	○	●	○	○	●	○	○	○	○	●	●	●	○	○	–	●	○	●	○	–	○	○	●
	Inter-agent messages	–	○	–	–	–	–	–	–	○	○	○	–	–	–	●	–	●	–	–	–	○	○	–	○	–	●	○	○	●	○	●
Evaluation	Deterministic grading	○	○	○	●	○	●	○	●	○	○	○	○	○	○	○	○	○	●	●	○	○	○	○	○	○	○	○	○	○	●	●
	Cost vs. a single agent	–	○	–	–	–	–	–	–	○	○	○	–	–	–	○	–	○	–	–	–	○	○	●	○	–	○	●	○	○	○	●
Threat	User is the attacker	●	○	○	○	○	●	○	–	○	○	○	●	○	●	○	●	○	○	●	●	○	●	–	○	●	○	●	○	○	○	○

Attacks specific to multi-agent systems. Defending agents from one another predates LLMs (Ramchurn et al., 2004) and is now framed as multi-agent security for LLM agents (Schroeder de Witt et al., 2025). Injected instructions self-replicate across connected agents (Lee et al., 2025; Gu et al., 2024), messages can be tampered in transit (He et al., 2025; Yan et al., 2026), and shared memory or documents poisoned (Chen et al., 2024b; Ju et al., 2026). Other work asks how the way agents are connected changes how far an injected instruction spreads (Yu et al., 2025; Zhu et al., 2025b; Li et al., 2026b; Huang et al., 2025; Khan et al., 2025). Benchmarks either fold robustness and task success into one score (Kavathekar et al., 2026), test a single architecture (Lee et al., 2026), or take the misuse model where the user is the attacker (Tur et al., 2025; Arora et al., 2025; Hagag et al., 2026; Chiang et al., 2025). ACIArena (An et al., 2026) measures cascading injection across teams, though not on agentic tasks.

What partitioning costs. A parallel literature prices coordination. Early gains from debate and role specialisation (Du et al., 2024; Qian et al., 2024; Hong et al., 2024; Wu et al., 2024; Wang et al., 2025; Qian et al., 2025; Tian et al., 2025) shrink once the single agent gets the same compute budget (Smit et al., 2024; Zhang et al., 2025a; Li et al., 2026c; Chen et al., 2024a), as base models improve (Gao et al., 2025), and at scale on tool-heavy tasks (Kim et al., 2025b). Their failures cluster into poor specification, misalignment between agents and weak verification (Cemri et al., 2025), and automatically designed teams (Hu et al., 2025; Zhang et al., 2025c) underperform chain-of-thought with self-consistency at higher cost (Jwalapuram et al., 2026). Peigne-Lefebvre et al. (2025) name the security side of this trade directly, setting hardening against collaboration capability as a "security tax".

What is new here. The pieces exist separately. Single-agent suites offer no topology to vary and no inter-agent channel to attack. Multi-agent studies build on non-agentic tasks, cover only part of the attack surface, or replicate the tools. The cost literature prices coordination among agents that each hold the whole toolbox. We price the complementary case, where no agent can finish alone, so the cost we report is the price of the partition itself.

3 THE MEASUREMENT ENVIRONMENT

SAFEMAS instantiates an agentic system in a simulated domain, a single agent or a team, and gives it a multi-part user task. The single agent holds the whole toolbox and can finish alone. In a team the tools are partitioned, so the task needs cooperation. In attacked scenarios we inject one payload at one position: in an agent's instructions, in a tool return, or in an inter-agent message.

3.1 DOMAINS, TOOLS, AND TASKS

Four domains (banking, DevOps, healthcare, and a workspace suite) provide a simulated environment and 240 tools: 188 reads, which inspect the state, and 52 writes, which change it. Each domain offers a different kind of write for an attacker to abuse. The read-heavy shape is deliberate. Nothing is handed to the agent: every value lives in the state and is reached by following references from one record to the next, so guessing fails, and an attacker who writes a value into a record uses the same channel the agent depends on (task validity, Zhu et al., 2025c).

Each domain defines six graded tasks (24 total) across three difficulty tiers. A task bundles three or four independent work streams, so every task can be split across a team. Difficulty is set by how many write operations a work stream requires, two, three or four, each reached through its own chain of reads.

3.2 ARCHITECTURES

We instantiate the five canonical structures of Kim et al. (2025b) at team sizes 3 and 4. Team size follows the task: each subagent owns one of its three or four work streams. Larger teams would need broader tasks, and each added agent further multiplies the model calls a run already spends. Figure 1 shows the four multi-agent structures. Tool-less agents, an orchestrator, consensus node or aggregator, only plan, route or merge. Only the subagents, which hold tools, can act. The architectures differ on whether an orchestrator directs the work and whether subagents talk to each other. Centralized has an orchestrator and no peer channel, decentralized has a peer channel and no orchestrator, and hybrid has both. Independent has neither. Its subagents work alone and a tool-less aggregator merges their reports at the end (Table 2). All do the same underlying work. Only who may act and who may speak to whom changes.

Table 2: Tool partition, who knows it, and the route to unowned reads, by architecture.

Family	Writes	Reads	Who knows the partition	Route to an unowned read
Single (S)	All	All	–	–
Centralized (C)	Partitioned	Partitioned	Orchestrator	Orchestrator relays
Hybrid (H)	Partitioned	Partitioned	Orchestrator	Orchestrator, then peer
Decentralized (D)	Partitioned	Partitioned	Every subagent	Peer
Independent (I)	Partitioned	Shared	–	None

3.3 PARTITIONING THE TOOLS BY CONSTRUCTION

Each domain's writes are partitioned into disjoint groups, one per subagent, and its reads partitioned evenly, with a separate partition per team size. A tool a subagent does not own is absent from its tool list. Two things follow. No subagent holds enough tools to finish alone, so delegation is required. And communication is forced: a subagent must ask a peer or the orchestrator for a value it cannot read, and every work stream requires at least one such request. The partition is the same in every team except the independent ones: with no channel to ask across, an unowned read would be unreachable and the task impossible, so each of their subagents holds every read. What varies is who knows it (Table 2), and that is deliberately not uniform: hand the ownership map to a hybrid subagent and it stops consulting the orchestrator, collapsing hybrid onto decentralized. Whether that variation helps or hurts is not obvious. Every added agent and message is a new place to inject and to forward a payload (Lee et al., 2025; He et al., 2025).

3.4 ATTACKS, AND WHERE THEY ARE PLACED

The attacker has one goal per scenario, a specific change to the environment state such as a transfer to their own account or a prescription rewritten to the wrong drug. They control one payload on one of three attack vectors (Figure 1): an agent's own instructions, a tool return, or a message tampered in transit. The tool and message vectors match attacks seen in practice: poisoned tool returns across public MCP servers (Zhao et al., 2026), and an attacker that rewrites A2A agent cards in transit (LevelBlue SpiderLabs, 2025).

A payload lands on one of three roles. The **target** agent owns the write the attack needs and is the only one that can carry it out. The **relay** agent pulls a value along the chain and, once lured, writes it into a record the target agent later reads. The **orchestrator** holds no tools and only delegates. It is present only in centralized and hybrid teams. Not every architecture carries every position: a payload reaches the orchestrator through its instructions or a message it routes, never a tool return.

Where the payload lands defines two kinds of attack. **Direct attacks** land on the target agent, which already holds the write, so nothing cascades and the only question is whether it complies. **Indirect attacks** land on a relay agent or the orchestrator and must cascade. Most are field redirects: the relay agent overwrites a record with a write it legitimately owns, and the target agent, following its resolution chain, reads the attacker's value and acts on it. Indirect attacks are what several agents make possible, and neither agent does anything its role forbids.

Each attack is performed at every injection position its architecture exposes. Four domains, the five architectures with each team run at two sizes, 24 tasks and 42 attacks yield 990 scenarios, 120 benign and 870 attacked, each run on every model.

4 METRICS

Every outcome comes from the post-run environment state, checked against the exact values the task required. No language model judges any of them, so each number is reproducible. A judge would cost more than it saves: its variance is large enough to swamp the effects measured (Feuer et al., 2025), it trusts what it already believes over what the context shows when the two disagree (Alloula et al., 2026), and it sits inside the attack surface, where framing flips its verdict (Raina et al., 2024; Gao, 2026).

Table 3: Task, attack, and coordination metrics. MAS: multi-agent team; SAS: single agent.

	Symbol	Metric	Definition	Interpretation
Task	U	Utility	writes landed/writes required	Work done, with partial credit
	W	Off-target writes	$\text{avg}(\#\text{ off-target writes})$	Off-target writes per run
Attack	D	Delivery	$\Pr(\text{reached an agent})$	How often the payload arrives
	ASR	Attack success	$\Pr(\text{attack succeeds})$	How often the attack lands
Coord.	T	Reasoning turns	# model calls in the run	Reasoning the team spends
	O	Overhead	$(T_{\text{MAS}} - T_{\text{SAS}})/T_{\text{SAS}}$	Turns beyond the single agent
	E_c	Efficiency	$U/(T/T_{\text{SAS}})$	Utility per unit of cost

The three blocks of Table 3 measure whether the work got done in the right records (task), whether an attack arrived and whether it succeeded (attack), and what the architecture costs with no attacker (coordination). The attack block separates whether a payload arrives from whether it succeeds: a design can look safe because payloads rarely arrive or because its agents refuse them. Utility is graded write by write: each required write that is present in the final state earns credit, so a stream can score partially. Off-target writes do not lower it, yet each changes the environment in a way the task did not ask for. We count them separately, excluding attacker-induced writes. Attack success is the fraction of attacked runs in which the attacker's value landed in the environment state.

5 RESULTS

We run every scenario on DeepSeek-V4-Flash, GPT-5.6 Luna and MiniMax-M2.7. The full grid of results is in Table 6 in Appendix A. The result is not the one a least-privilege argument predicts: in the orchestrated teams, partitioning the tools does not make the target agent harder to attack, and the channels it creates add attack surface. In the single agent, the relay and target roles are played by the same agent, so an indirect attack cascades through its own records. It writes the attacker's value while doing the relay job and reads it back when it reaches the target one.

5.1 ATTACK SUCCESS ACROSS ARCHITECTURES

When the single agent and the teams face the same attacks, partitioning buys little. Across the three models these attacks succeed on the single agent in 41% of runs, on the hybrid teams in 40% and on the centralized teams in 39%, against 32% for the decentralized teams and 19% for the independent ones. The boundary an orchestrator interposes between the poisoned value and the target agent stops attacks on one model only. Over all the attacks they face, the orchestrated teams are attacked more often than the single agent, 53% against 41% (Table 4, Figure 2a), and the channels an orchestrator adds account for the whole difference (§5.2). Structure changes which agent the attacker must reach and how far the payload must travel, not how often an agent obeys the payload once it arrives.

Table 4: Results by architecture and model. U with no attack, U_{atk} under attack.

	DeepSeek-V4-Flash				GPT-5.6 Luna				MiniMax-M2.7			
Architecture	U	U_{atk}	$\text{ASR}_{\text{direct}}$	$\text{ASR}_{\text{indirect}}$	U	U_{atk}	$\text{ASR}_{\text{direct}}$	$\text{ASR}_{\text{indirect}}$	U	U_{atk}	$\text{ASR}_{\text{direct}}$	$\text{ASR}_{\text{indirect}}$
Single	97	96	19	42	82	81	35	62	92	89	44	44
Centralized	54	52	24	30	50	50	70	71	21	27	63	59
Hybrid	62	63	30	30	54	51	64	72	29	31	64	57
Decentralized	86	83	19	23	48	54	29	23	54	56	60	39
Independent	80	80	7	9	61	64	18	17	81	78	28	35

Cascading attacks succeed on the single agent on all three models, up to 62% on GPT-5.6 Luna, since nothing interrupts a cascade that runs inside one agent. On GPT-5.6 Luna and MiniMax-M2.7 the orchestrated designs are attacked more often and complete far less of the work, so they pay the price of coordination and get no protection in return. On DeepSeek-V4-Flash they are attacked less, but they still complete far less of the work. The independent teams are the least attacked on all three models, and they complete nearly as much work as the single agent. Their agents cascade through the shared records like everyone else's, but each handles only its own stream, so an injected write more often falls outside its scope. The value lands in a record in a fifth of indirect runs, against nearly half for the single agent, which handles every stream.

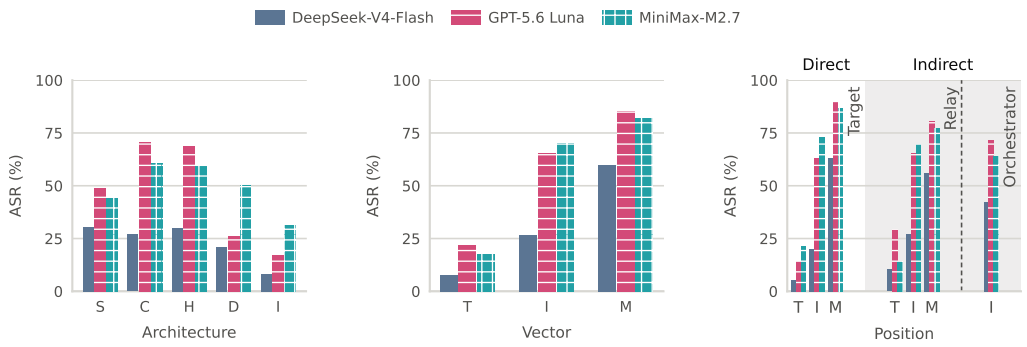
GPT-5.6 Luna and MiniMax-M2.7 are attacked more often than DeepSeek-V4-Flash in every architecture. A third of the attacks fail on all three models, one in six succeeds on all three, and half succeed on some models and not others. A successful attack also costs the task nothing. Utility under attack stays within a few points of benign utility in every architecture on all three models (Table 4), and runs where the attack succeeded complete as much of the task as benign runs. This is because most successful attacks add an entry, such as a transfer, next to the writes of the task, and only a few overwrite a value the task needs. The injected action is one more step in a plan the agent was already executing, so nothing in task completion reveals it.

Takeaway (RQ1). More agents is not more security: attacks on the orchestrated teams succeed as often as on the single agent when both face the same attack, and more often once the attacks on inter-agent channels are added. A successfully attacked run completes its task as well as a benign one, so task completion cannot detect an attack.

5.2 ATTACK VECTORS AND POSITIONS

Where the malicious payload lands matters more than how the team is wired. The inter-agent message is the most dangerous attack vector: it succeeds in 76% of runs across the three models, and up to 85% on GPT-5.6 Luna (Figure 2b). Injected instructions follow at 54%. A poisoned tool return

is the weakest vector at 16%, and as low as 8% on DeepSeek-V4-Flash. The ranking is the same on all three models. Depending on where the payload lands, attacks succeed in 14% to 80% of runs, against 19% to 53% depending on the architecture.



(a) Attack success by architecture. (b) Attack success by vector. (c) By position.

Figure 2: Attack success on DeepSeek-V4-Flash, GPT-5.6 Luna and MiniMax-M2.7. In (b,c), T tool return, I agent instructions, M orchestrator message tampered in transit.

The orchestrator adds three positions, the orchestrator's own instructions and the messages it sends to the relay and to the target agent. Across the three models they are the three most dangerous positions in the benchmark (Figure 2c), and the two tampered messages are the most dangerous on each of them. The tampered message to the target agent is the most effective: the attacker owns the channel, and the message is followed in nearly two of three runs on DeepSeek-V4-Flash and in nine of ten on GPT-5.6 Luna. Out of the 149 scenarios that succeed on all three models, 109 are orchestrated teams and 67 are tampered-message positions. Next come the tampered message to a relay agent, at 71%, and the orchestrator's own instructions, at 59%. A payload placed there succeeds one and a half times as often as the same payload placed at a relay agent on DeepSeek-V4-Flash, and about as often on the other two models, although the orchestrator holds no tool and must persuade a subagent to act. The orchestrator restates the payload in its own words when it delegates the stream, so the subagent receives it as an instruction from its orchestrator, with no sign of the record it came from. The orchestrator is where provenance is lost. Removing its tools relocates the danger rather than removing it, because the routing is itself a capability.

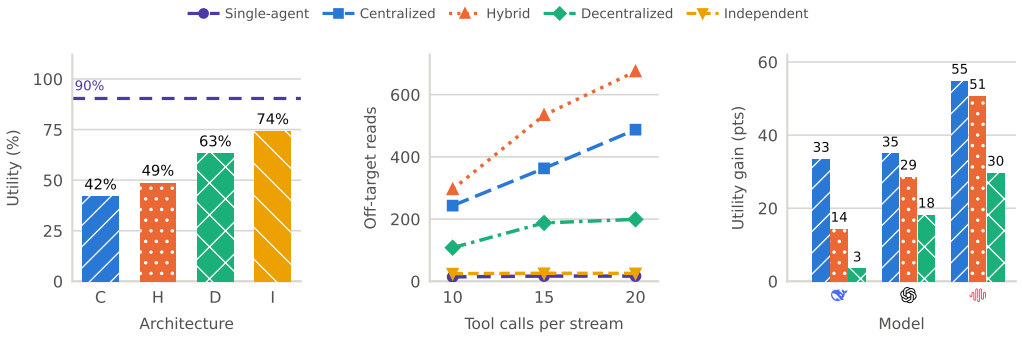
Delivery does not explain the differences between positions. The payload reaches an agent in every run for injected instructions and messages, and in nearly every run for tool returns (Table 6). What separates them is what the receiving agent does with it. A payload inside a tool return is read as data and competes with the task's own data: agents follow it in about one delivered run in six. Injected instructions are followed in about half of the delivered runs, at the target agent and at the relay agent alike. A message from the orchestrator is followed in more than seven of ten. When an architecture blocks an attack, it often does so by tool absence rather than judgment. In one GPT-5.6 Luna centralized run the tampered message tells a subagent that owns a different stream to schedule a payment to the attacker's account, and it answers that it holds neither the payment tool nor the records the payment needs. It wanted to comply and could not. This is what gives the orchestrated teams their low rate on DeepSeek-V4-Flash, and it protects nothing once the payload reaches an agent that holds the tool.

Takeaway (RQ2). Secure the channel before the architecture: a tampered inter-agent message is the most effective attack vector on all three models, and it exists only because the team exists. The payload almost always arrives, so what decides the outcome is whether the receiving agent complies, and it complies most with a message from the orchestrator.

5.3 The cost of partitioning the tools

With no attacker present, the 120 benign scenarios price the team. Only the single agent can finish a task alone, so a team's utility gap against it combines two costs: the partition and coordination.

This gap is large on all three models, and taken together it is largest where an orchestrator sits between the subagents (Figure 3a, Table 4). The single agent completes the most, the independent teams stay within reach of it, and the centralized and hybrid teams trail on all three models, down to under a quarter of the single agent's work for the centralized teams on MiniMax. The lost effort goes into reads the task never asked for (Figure 3b). On the longest work streams, those that take 20 tool calls to complete, the two orchestrated architectures issue 490 and 670 such reads per run, against about 20 for the single agent and 25 for the independent teams. The same architectures also make the most off-target writes, 2.5 and 3.4 per benign run against 0.8 for the single agent, and values relayed through the orchestrator come back wrong. In one benign MiniMax run an invoice is reported as paid at more than 15 times its amount.



(a) Utility by architecture. (b) Off-target reads per run. (c) Utility gain with shared reads.

Figure 3: Benign-run utility, off-target reads, and the utility gain when reads are shared.

To separate the cost of the partition from the cost of the team, we rerun the centralized, hybrid and decentralized teams with one change: every subagent can read everything, while writes stay partitioned and the coordination protocol stays the same (Figure 3c, Table 5). The utility recovered is the cost of the partition. Whatever gap remains to the single agent is the cost of coordination. For the centralized teams the partition accounts for most of the cost on all three models: sharing the reads returns 33 to 55 points. On GPT-5.6 Luna the hybrid teams then complete as much as the single agent. On MiniMax the two orchestrated architectures climb from under half of the single agent's utility to above four-fifths of it once every subagent can see the data. The partition costs the decentralized teams the least, since they have no orchestrator to relay values through. The little protection the partition gives is therefore bought with a capability loss that a shared read layer would largely refund.

Table 5: The coordination metrics with no attacker present, per model and architecture.

Model	DeepSeek								Luna								MiniMax							
Arch.	S	C		H		D		I	S	C		H		D		I	S	C		H		D		I
Reads	all	own	all	own	all	own	all	all	all	own	all	own	all	own	all	all	all	own	all	own	all	own	all	all
T	13	211	123	439	123	137	48	41	7	344	73	410	76	157	25	28	7	244	60	219	38	227	31	30
O	–	15.3	8.5	32.9	8.5	9.6	2.7	2.1	–	48.1	9.4	57.6	9.8	21.5	2.6	2.9	–	34.0	7.7	30.5	4.4	31.6	3.4	3.3
E_c	0.97	0.03	0.09	0.02	0.08	0.08	0.24	0.26	0.82	0.01	0.08	0.01	0.08	0.02	0.19	0.16	0.92	0.01	0.09	0.01	0.15	0.02	0.19	0.19

The coordination metrics show where the effort goes (Table 5). With partitioned reads a centralized or hybrid team spends 16 to 60 times the single agent's reasoning turns, a decentralized team 11 to 33, an independent team 3 to 4. The extra turns buy no extra task completion: efficiency ranks the independent teams first and the orchestrated teams last on all three models. Sharing the reads removes most of those turns: the centralized and hybrid teams fall to between 5 and 11 times the single agent's turns, and their efficiency rises between 3 and 16 fold.

Takeaway (RQ3). Partitioning creates complexity, and the complexity is the cost: the architectures with an orchestrator complete about half of the single agent's work, a quarter on the weakest model,

and spend an order of magnitude more reasoning turns re-fetching data the partition hid from them. Sharing the reads returns most of that loss, so the team pays for the partition without receiving its safety.

6 Discussion and limitations

Architecture changes how often attacks succeed but never stops them, so what remains falls to defences. Our results point to three directions:

Watch the data, not just the messages. Message-level defences (Syros et al., 2026; Shi et al., 2026; Dai et al., 2026) check what agents send one another, and even these can be bypassed (Jha et al., 2026). But an attack through state sends no message: one agent writes the attacker's value into a record, and another reads it later while doing its own job. The attack vector is a shared record, not a conversation, and a proper defence should watch the state-dependency graph those records form.

Tell a benign change from a malicious one. A record does not say who last wrote it, and the agent that uses it has no reason to distrust it, so per-agent defences (Cai et al., 2026) do not catch a value that one agent writes and another later reads. What is missing is a mechanism to tell whether the state an agent reads is still untampered, so a change is trusted or refused before it is acted on.

Separate orchestration from execution in parallel teams. Our orchestrator re-plans from what its subagents report. Other designs fix the plan before any data is read and keep executors apart (Debenedetti et al., 2026; Li et al., 2026a; Kim et al., 2025a; Wu et al., 2025; Bagdasarian et al., 2024). What is missing is a design that keeps this separation in teams whose agents work in parallel and message their peers.

Limitations: Five choices limit the scope of our work: (1) *Sampling.* Each scenario runs once per model, and we report point estimates; repeated runs, which would bound the model's own stochasticity, are costly, since one partitioned-team scenario takes hundreds of model calls. (2) *Attacker.* The attacker is weak by construction, one payload at one position with no rephrasing or combined channels, so every rate is a floor that adaptive attackers beat (Nasr et al., 2026) (3) *Prompts.* Each architecture comes with its own role prompts, so its effect cannot be separated from their wording. (4) *Defences.* We test bare architectures, with no defence (Hines et al., 2024; Wallace et al., 2024; Chen et al., 2025; Zhu et al., 2025a; Shi et al., 2025; Costa et al., 2025). (5) *Partition and damage.* Tools are partitioned evenly rather than grouped by what they do, and every successful attack counts the same whatever its damage. The results therefore say whether partitioning changes exposure, not how to partition well or which successful attacks hurt most.

7 Conclusion

Separation of privilege suggests a remedy for prompt injection: partition the tools across a team of agents, so that the agent an attacker reaches is not the agent that can act. Over 990 scenarios, each run on three models across five architectures, the principle does not hold for orchestrated teams. More agents is not more security. The partition does not protect the agent that can act, and the channels it creates add new ways in: attacks on orchestrated teams succeed more often than on the single agent. The attack vector matters more than the wiring: a tampered inter-agent message is the most effective attack vector on all three models. Secure that channel first. The one structure that helps is the one that partitions least: teams whose agents share every read and never message one another are the least attacked. And the partition is expensive for the wrong reason. The orchestrated teams lose half of the single agent's task completion re-fetching data the partition hid from them, and most of it comes back when the reads are shared. On this benchmark, the safety argument for multi-agent partitioning does not survive contact with the measurement.

AI use statement

We used generative AI tools to edit the manuscript for readability and consistency, and to assist in writing the framework code and in generating the simulated environments, tasks, and attack payloads. We have not used generative AI tools to propose the hypotheses or design the methodology.

We have reviewed all AI-assisted work: the code and environments were validated by the authors, the graders are deterministic, and every reported number is recomputable from the released traces. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work releases working prompt-injection payloads. All attacks execute only against simulated environments with synthetic data: no deployed system, real account, or personal record is involved, and each payload names records that exist only in those environments, so none transfers to a real target as written. The intent of the release is defensive, to let architecture and defence choices be measured before deployment. No human subjects were involved.

REPRODUCIBILITY STATEMENT

All results are reproducible from the released artifact at `https://anonymous.4open.science/r/safemas-6946/`: the four environments, the 990 scenario specifications, the role prompts for every architecture, and the scripts that compute every table and figure. The run traces are archived separately at `https://osf.io/xfrzh/overview?view_only=d3ef65c997204c1b97340591be471019` (SHA-256 `0a4566ac...a946`). Grading is deterministic from the post-run environment state, with no LLM judge, so every number in Section 5 can be recomputed from the traces alone. The three models were called through their public APIs under the identifiers `deepseek-v4-flash` (`api.deepseek.com`), `gpt-5.6-luna` (`api.openai.com`) and `MiniMax-M2.7` (`api.minimax.io`). All campaigns, including the no-partition control of Section 5.3, ran in September 2026.

REFERENCES

Anissa Alloula, Federico Licini, Ava Batchkala, and Seraphina Goldfarb-Tarrant. Safety is contextual, LLM-judges are not: Navigating the rigid priors of evaluators. *arXiv preprint arXiv:2606.07874*, 2026.

Hengyu An, Minxi Li, Jinghuai Zhang, Naen Xu, Chunyi Zhou, Changjiang Li, Xiaogang Xu, Tianyu Du, and Shouling Ji. ACIArena: Toward unified evaluation for agent cascading injection. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2026. arXiv:2604.07775.

Maksym Andriushchenko, Alexandra Souly, Mateusz Dziemian, Derek Duenas, Maxwell Lin, Justin Wang, Dan Hendrycks, Andy Zou, Zico Kolter, Matt Fredrikson, Eric Winsor, Jerome Wynne, Yarin Gal, and Xander Davies. AgentHarm: A benchmark for measuring harmfulness of LLM agents. In *International Conference on Learning Representations (ICLR)*, 2025.

Anthropic. Introducing the model context protocol. `https://www.anthropic.com/news/model-context-protocol`, 2024. Accessed 2026-08-16.

Anthropic. Building agents with the Claude Agent SDK. `https://claude.com/blog/building-agents-with-the-claude-agent-sdk`, 2025. Accessed 2026-08-16.

Nirmit Arora, Sathvik Joel, Ishan Kavathekar, Palak, Rohan Gandhi, Yash Pandya, Tanuja Ganu, Aditya Kanade, and Akshay Nambi. Exposing weak links in multi-agent systems under adversarial prompting. *arXiv preprint arXiv:2511.10949*, 2025.

Eugene Bagdasarian, Ren Yi, Sahra Ghalebikesabi, Peter Kairouz, Marco Gruteser, Sewoong Oh, Borja Balle, and Daniel Ramage. AirGapAgent: Protecting privacy-conscious conversational agents. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2024. arXiv:2405.05175.

Andrea Bittau, Petr Marchenko, Mark Handley, and Brad Karp. Wedge: Splitting applications into reduced-privilege compartments. In *5th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2008.

Leo Boisvert, Mihir Bansal, Chandra Kiran Reddy Evuru, Gabriel Huang, Abhay Puri, Avinandan Bose, Maryam Fazel, Quentin Cappart, Jason Stanley, Alexandre Lacoste, Alexandre Drouin, and Krishnamurthy Dvijotham. DoomArena: A framework for testing ai agents against evolving security threats. *arXiv preprint arXiv:2504.14064*, 2025.

Yuandao Cai, Wensheng Tang, Cheng Wen, and Shengchao Qin. Ghost in the agent: Redefining information flow tracking for LLM agents. *arXiv preprint arXiv:2604.23374*, 2026.

Mert Cemri, Melissa Z. Pan, Shuyi Yang, Lakshya A. Agrawal, Bhavya Chopra, Rishabh Tiwari, Kurt Keutzer, Aditya Parameswaran, Dan Klein, Kannan Ramchandran, Matei Zaharia, Joseph E. Gonzalez, and Ion Stoica. Why do multi-agent LLM systems fail? In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2025. arXiv:2503.13657.

Lingjiao Chen, Jared Quincy Davis, Boris Hanin, Peter Bailis, Ion Stoica, Matei Zaharia, and James Zou. Are more LLM calls all you need? towards the scaling properties of compound AI systems. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024a. arXiv:2403.02419.

Sizhe Chen, Julien Piet, Chawin Sitawarin, and David Wagner. StruQ: Defending against prompt injection with structured queries. In *34th USENIX Security Symposium (USENIX Security)*, 2025. arXiv:2402.06363.

Zhaorun Chen, Zhen Xiang, Chaowei Xiao, Dawn Song, and Bo Li. AgentPoison: Red-teaming LLM agents via poisoning memory or knowledge bases. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024b.

Jeffrey Yang Fan Chiang, Seungjae Lee, Jia-Bin Huang, Furong Huang, and Yizheng Chen. Why are web AI agents more vulnerable than standalone LLMs? a security analysis. *arXiv preprint arXiv:2502.20383*, 2025.

Manuel Costa, Boris Köpf, Aashish Kolluri, Andrew Paverd, Mark Russinovich, Ahmed Salem, Shruti Tople, Lukas Wutschitz, and Santiago Zanella-Béguelin. Securing AI agents with information-flow control. *arXiv preprint arXiv:2505.23643*, 2025.

CrewAI. CrewAI: Framework for orchestrating role-playing, autonomous AI agents. `https://github.com/crewAIInc/crewAI`, 2024. Accessed 2026-08-16.

Haowen Dai, Zonghao Ying, Wenfeng Li, Xiangfan Wu, Yisong Xiao, Tianyuan Zhang, Jiaye Lin, Lei Wei, Guangyuan Dong, Xitong Ling, Xixun Lin, Quanchen Zou, and Xiangzheng Zhang. SafeFlow: Semantic information-flow control for blocking malicious propagation in multi-agent systems. *arXiv preprint arXiv:2607.25255*, 2026.

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 37 (NeurIPS)*, 2024. arXiv:2406.13352.

Edoardo Debenedetti, Ilia Shumailov, Tianqi Fan, Jamie Hayes, Nicholas Carlini, Daniel Fabian, Christoph Kern, Chongyang Shi, Andreas Terzis, and Florian Tramèr. Defeating prompt injections by design. In *IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*, 2026. arXiv:2503.18813.

Santiago Díaz, Christoph Kern, and Kara Olive. An introduction to Google's approach to AI agent security. `https://research.google/pubs/an-introduction-to-googles-approach-for-secure-ai-agents/`, 2025. Accessed 2026-08-16.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. In *International Conference on Machine Learning (ICML)*, 2024. arXiv:2305.14325.

Ivan Evtimov, Arman Zharmagambetov, Aaron Grattafiori, Chuan Guo, and Kamalika Chaudhuri. WASP: Benchmarking web agent security against prompt injection attacks. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2025. arXiv:2504.18575.

Benjamin Feuer, Chiung-Yi Tseng, Astitwa Sarthak Lathe, Oussama Elachqar, and John P. Dickerson. When judgment becomes noise: How design failures in LLM judge benchmarks silently undermine validity. *arXiv preprint arXiv:2509.20293*, 2025.

Mingyan Gao, Yanzi Li, Banruo Liu, Yifan Yu, Phillip Wang, Ching-Yu Lin, and Fan Lai. Single-agent or multi-agent systems? Why not both? *arXiv preprint arXiv:2505.18286*, 2025.

Yang Gao. How reliable is your jailbreak judge? calibration and adversarial robustness of automated ASR scoring. *arXiv preprint arXiv:2606.25487*, 2026.

Gartner. Gartner predicts over 40% of agentic AI projects will be canceled by end of 2027. `https://www.gartner.com/en/newsroom/press-releases/2025-06-25-gartner-predicts-over-40-percent-of-agentic-ai-projects-will-be-canceled-by-end-of-2027`, 2025. Accessed 2026-08-16.

Google. Announcing the Agent2Agent protocol (A2A). `https://developers.googleblog.com/en/a2a-a-new-era-of-agent-interoperability/`, 2025a. Accessed 2026-08-16.

Google. Agent development kit: Making it easy to build multi-agent applications. `https://developers.googleblog.com/en/agent-development-kit-easy-to-build-multi-agent-applications/`, 2025b. Accessed 2026-08-16.

Kai Greshake, Sahar Abdelnabi, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario Fritz. Not what you've signed up for: Compromising real-world LLM-integrated applications with indirect prompt injection. In *Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security (AISec)*, 2023. arXiv:2302.12173.

Xiangming Gu, Xiaosen Zheng, Tianyu Pang, Chao Du, Qian Liu, Ye Wang, Jing Jiang, and Min Lin. Agent smith: A single image can jailbreak one million multimodal LLM agents exponentially fast. In *International Conference on Machine Learning (ICML)*, 2024. arXiv:2402.08567.

Chengquan Guo, Xun Liu, Chulin Xie, Andy Zhou, Yi Zeng, Zinan Lin, Dawn Song, and Bo Li. RedCode: Risky code execution and generation benchmark for code agents. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2024. arXiv:2411.07781.

Ben Hagag, William L. Anderson, Christian Schroeder de Witt, and Sarah Scheffler. Architecture matters for multi-agent security. In *International Conference on Machine Learning (ICML)*, 2026. arXiv:2604.23459.

Pengfei He, Yupin Lin, Shen Dong, Han Xu, Yue Xing, and Hui Liu. Red-teaming LLM multi-agent systems via communication attacks. In *Findings of the Association for Computational Linguistics: ACL 2025*, 2025. arXiv:2502.14847.

Keegan Hines, Gary Lopez, Matthew Hall, Federico Zarfati, Yonatan Zunger, and Emre Kiciman. Defending against indirect prompt injection attacks with spotlighting. *arXiv preprint arXiv:2403.14720*, 2024.

Sirui Hong, Mingchen Zhuge, Jiaqi Chen, Xiawu Zheng, Yuheng Cheng, Ceyao Zhang, Jinlin Wang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. MetaGPT: Meta programming for a multi-agent collaborative framework. In *International Conference on Learning Representations (ICLR)*, 2024. arXiv:2308.00352.

Shengran Hu, Cong Lu, and Jeff Clune. Automated design of agentic systems. In *International Conference on Learning Representations (ICLR)*, 2025. arXiv:2408.08435.

Jen-tse Huang, Jiaxu Zhou, Tailin Jin, Xuhui Zhou, Zixi Chen, Wenxuan Wang, Youliang Yuan, Michael R. Lyu, and Maarten Sap. On the resilience of LLM-based multi-agent collaboration with faulty agents. In *International Conference on Machine Learning (ICML)*, 2025. arXiv:2408.00989.

Rishi Jha, Harold Triedman, Justin Wagle, and Vitaly Shmatikov. Breaking and fixing defenses against control-flow hijacking in multi-agent systems. In *International Conference on Learning Representations (ICLR)*, 2026. arXiv:2510.17276.

Tianjie Ju, Yiting Wang, Xinbei Ma, Pengzhou Cheng, Haodong Zhao, Yulong Wang, Lifeng Liu, Jian Xie, Zhuosheng Zhang, and Gongshen Liu. Flooding spread of manipulated knowledge in LLM-based multi-agent communities. *Science China Information Sciences*, 69(7), 2026. arXiv:2407.07791.

Prathyusha Jwalapuram, Hehai Lin, Chuyuan Li, Fangkai Jiao, Sudong Wang, Yifei Ming, Zixuan Ke, Chengwei Qin, Giuseppe Carenini, and Shafiq Joty. The illusion of multi-agent advantage. *arXiv preprint arXiv:2606.13003*, 2026.

Ishan Kavathekar, Hemang Jain, Ameya Rathod, Ponnurangam Kumaraguru, and Tanuja Ganu. TAMAS: Benchmarking adversarial risks in multi-agent LLM systems. In *Proceedings of the Association for Computational Linguistics (ACL)*, 2026. arXiv:2511.05269.

Rana Muhammad Shahroz Khan, Zhen Tan, Sukwon Yun, Charles Fleming, and Tianlong Chen. Agents under siege: Breaking pragmatic multi-agent LLM systems with optimized prompt attacks. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2025. arXiv:2504.00218.

Juhee Kim, Woohyuk Choi, and Byoungyoung Lee. Prompt flow integrity to prevent privilege escalation in LLM agents. *arXiv preprint arXiv:2503.15547*, 2025a.

Juhee Kim, Xiaoyuan Liu, Zhun Wang, Shi Qiu, Bo Li, Wenbo Guo, and Dawn Song. SoK: Attack and defense landscape of agentic AI systems. In *35th USENIX Security Symposium (USENIX Security)*, 2026. Extended version: arXiv:2603.11088.

Yubin Kim, Ken Gu, Chanwoo Park, Chunjong Park, Samuel Schmidgall, A. Ali Heydari, Yao Yan, Zhihan Zhang, Yuchen Zhuang, Yun Liu, Mark Malhotra, Paul Pu Liang, Hae Won Park, Yuzhe Yang, Xuhai Xu, Yilun Du, Shwetak Patel, Tim Althoff, Daniel McDuff, and Xin Liu. Towards a science of scaling agent systems. *arXiv preprint arXiv:2512.08296*, 2025b.

Thomas Kuntz, Agatha Duzan, Hao Zhao, Francesco Croce, Zico Kolter, Nicolas Flammarion, and Maksym Andriushchenko. OS-Harm: A benchmark for measuring safety of computer use agents. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2025. arXiv:2506.14866.

LangChain. LangGraph. `https://www.langchain.com/langgraph`, 2024. Accessed 2026-08-16.

Donghyun Lee, Mo Tiwari, and Brando Miranda. Prompt infection: LLM-to-LLM prompt injection within multi-agent systems. In *Computer Security – ESORICS 2025 International Workshops*, 2025. arXiv:2410.07283.

Hanwool Lee, Dasol Choi, Bokyeong Kim, et al. NRT-Bench: Benchmarking multi-turn red-teaming of LLM operator agents in safety-critical control rooms. *arXiv preprint arXiv:2606.20408*, 2026.

LevelBlue SpiderLabs. Agent in the middle: Abusing agent cards in the agent-to-agent (A2A) protocol to win all the tasks. `https://www.levelblue.com/blogs/spiderlabs-blog/agent-in-the-middle-abusing-agent-cards-in-the-agent-2-agent-protocol-to-win-all-the-tasks`, 2025. Accessed 2026-08-16.

Evan Li, Tushin Mallick, Evan Rose, William Robertson, Alina Oprea, and Cristina Nita-Rotaru. ACE: A security architecture for LLM-integrated app systems. In *Network and Distributed System Security Symposium (NDSS)*, 2026a. arXiv:2504.20984.

Kunyang Li, Kyle Domico, Jonathan Gregory, and Patrick McDaniel. MESA: Prioritizing vulnerable communication channels for securing multi-agent systems. *arXiv preprint arXiv:2606.30602*, 2026b.

Wenzhe Li, Yong Lin, Mengzhou Xia, and Chi Jin. Rethinking mixture-of-agents: Is mixing different large language models beneficial? *Transactions on Machine Learning Research*, 2026c. arXiv:2502.00674.

Yupei Liu, Yuqi Jia, Runpeng Geng, Jinyuan Jia, and Neil Zhenqiang Gong. Formalizing and benchmarking prompt injection attacks and defenses. In *33rd USENIX Security Symposium (USENIX Security)*, 2024. arXiv:2310.12815.

Hanjun Luo, Shenyu Dai, Chiming Ni, Xinfeng Li, Guibin Zhang, Kun Wang, Tongliang Liu, and Hanan Salam. AgentAuditor: Human-level safety and security evaluation for LLM agents. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. arXiv:2506.00641.

Milad Nasr, Nicholas Carlini, Chawin Sitawarin, Sander Schulhoff, Jamie Hayes, Michael Ilie, Juliette Pluto, Shuang Song, Harsh Chaudhari, Ilia Shumailov, Abhradeep Thakurta, Kai Yuanqing Xiao, Andreas Terzis, and Florian Tramèr. The attacker moves second: Stronger adaptive attacks bypass defenses against LLM jailbreaks and prompt injections. In *35th USENIX Security Symposium (USENIX Security)*, 2026. arXiv:2510.09023.

NVD. CVE-2025-32711: M365 Copilot information disclosure (EchoLeak). `https://nvd.nist.gov/vuln/detail/CVE-2025-32711`, 2025a. Zero-click indirect prompt injection, CVSS 9.3. Accessed 2026-08-16.

NVD. CVE-2025-68664: LangChain serialization injection (LangGrinch). `https://nvd.nist.gov/vuln/detail/CVE-2025-68664`, 2025b. User data deserialized as trusted LangChain objects, CVSS 8.2. Accessed 2026-08-24.

NVD. CVE-2025-49596: Anthropic MCP Inspector remote code execution. `https://nvd.nist.gov/vuln/detail/CVE-2025-49596`, 2025c. Unauthenticated RCE over the Inspector proxy, CVSS 9.4. Accessed 2026-08-24.

NVD. CVE-2025-6514: mcp-remote OS command injection. `https://nvd.nist.gov/vuln/detail/CVE-2025-6514`, 2025d. Command injection from a malicious authorization endpoint, CVSS 9.6. Accessed 2026-08-24.

NVD. CVE-2026-13341: Kong Konnect MCP server indirect prompt injection. `https://nvd.nist.gov/vuln/detail/CVE-2026-13341`, 2026. Indirect prompt injection drives unintended API requests, CVSS 7.4. Accessed 2026-08-24.

OpenAI. New tools for building agents. `https://openai.com/index/new-tools-for-building-agents/`, 2025. Agents SDK. Accessed 2026-08-16.

OpenAI. OpenAI and Hugging Face partner to address security incident during model evaluation. `https://openai.com/index/hugging-face-model-evaluation-security-incident/`, 2026. OpenAI models compromised Hugging Face production infrastructure during a benchmark evaluation. Accessed 2026-08-27.

OWASP. OWASP top 10 for LLM applications 2025. `https://genai.owasp.org/llm-top-10/`, 2025. LLM01:2025 Prompt Injection. Accessed 2026-08-16.

Pierre Peigne-Lefebvre, Mikolaj Kniejski, Filip Sondej, Matthieu David, Jason Hoelscher-Obermaier, Christian Schroeder de Witt, and Esben Kran. Multi-agent security tax: Trading off security and collaboration capabilities in multi-agent systems. In *Proceedings of the Thirty-Ninth AAAI Conference on Artificial Intelligence (AAAI)*, 2025. arXiv:2502.19145.

Fábio Perez and Ian Ribeiro. Ignore previous prompt: Attack techniques for language models. In *NeurIPS ML Safety Workshop*, 2022. arXiv:2211.09527.

Niels Provos, Markus Friedl, and Peter Honeyman. Preventing privilege escalation. In *12th USENIX Security Symposium (USENIX Security)*, 2003.

Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, Juyuan Xu, Dahai Li, Zhiyuan Liu, and Maosong Sun. Chatdev: Communicative agents for software development. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024. arXiv:2307.07924.

Chen Qian, Zihao Xie, Yifei Wang, Wei Liu, Kunlun Zhu, Hanchen Xia, Yufan Dang, Zhuoyun Du, Weize Chen, Cheng Yang, Zhiyuan Liu, and Maosong Sun. Scaling large language model-based multi-agent collaboration. In *International Conference on Learning Representations (ICLR)*, 2025. arXiv:2406.07155.

Vyas Raina, Adian Liusie, and Mark Gales. Is LLM-as-a-judge robust? investigating universal adversarial attacks on zero-shot LLM assessment. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2024. arXiv:2402.14016.

Sarvapali D. Ramchurn, Dong Huynh, and Nicholas R. Jennings. Trust in multi-agent systems. *The Knowledge Engineering Review*, 19(1):1–25, 2004.

Yangjun Ruan, Honghua Dong, Andrew Wang, Silviu Pitis, Yongchao Zhou, Jimmy Ba, Yann Dubois, Chris J. Maddison, and Tatsunori Hashimoto. Identifying the risks of LM agents with an LM-emulated sandbox. In *International Conference on Learning Representations (ICLR)*, 2024. arXiv:2309.15817.

Jerome H. Saltzer and Michael D. Schroeder. The protection of information in computer systems. *Proceedings of the IEEE*, 63(9):1278–1308, 1975.

Christian Schroeder de Witt et al. Open challenges in multi-agent security: Towards secure systems of interacting AI agents. *arXiv preprint arXiv:2505.02077*, 2025.

Ruxue Shi, Yili Wang, Mengnan Du, Qinggang Zhang, Rui Miao, Yixin Liu, and Xin Wang. SAI-Guard: Communication-state simulation for proactive defense of LLM multi-agent systems. *arXiv preprint arXiv:2606.12474*, 2026.

Tianneng Shi, Jingxuan He, Zhun Wang, Hongwei Li, Linyu Wu, Wenbo Guo, and Dawn Song. Progent: Securing AI agents with privilege control. *arXiv preprint arXiv:2504.11703*, 2025.

Andries Smit, Nathan Grinsztajn, Paul Duckworth, Thomas D. Barrett, and Arnu Pretorius. Should we be going MAD? a look at multi-agent debate strategies for LLMs. In *International Conference on Machine Learning (ICML)*, 2024. arXiv:2311.17371.

Georgios Syros, Anshuman Suri, Jacob Ginesin, Cristina Nita-Rotaru, and Alina Oprea. SAGA: A security architecture for governing AI agentic systems. In *Network and Distributed System Security Symposium (NDSS)*, 2026. arXiv:2504.21034.

Aaron Xuxiang Tian, Ruofan Zhang, Jiayao Tang, Young Min Cho, Xueqian Li, Qiang Yi, Ji Wang, Zhunping Zhang, Danrui Qi, Zekun Li, Xingyu Xiang, Sharath Chandra Guntuku, Lyle Ungar, Tianyu Shi, and Chi Wang. Beyond the strongest LLM: Multi-turn multi-agent orchestration vs. single LLMs on benchmarks. *arXiv preprint arXiv:2509.23537*, 2025.

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

Sanidhya Vijayvargiya, Aditya Bharat Soni, Xuhui Zhou, Zora Zhiruo Wang, Nouha Dziri, Graham Neubig, and Maarten Sap. OpenAgentSafety: A comprehensive framework for evaluating real-world AI agent safety. In *International Conference on Learning Representations (ICLR)*, 2026. arXiv:2507.06134.

Eric Wallace, Kai Xiao, Reimar Leike, Lilian Weng, Johannes Heidecke, and Alex Beutel. The instruction hierarchy: Training LLMs to prioritize privileged instructions. *arXiv preprint arXiv:2404.13208*, 2024.

Junlin Wang, Jue Wang, Ben Athiwaratkun, Ce Zhang, and James Zou. Mixture-of-agents enhances large language model capabilities. In *International Conference on Learning Representations (ICLR)*, 2025. arXiv:2406.04692.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W. White, Doug Burger, and Chi Wang. AutoGen: Enabling next-gen LLM applications via multi-agent conversation. In *Conference on Language Modeling (COLM)*, 2024. arXiv:2308.08155.

Yuhao Wu, Franziska Roesner, Tadayoshi Kohno, Ning Zhang, and Umar Iqbal. IsolateGPT: An execution isolation architecture for LLM-based agentic systems. In *Network and Distributed System Security Symposium (NDSS)*, 2025. arXiv:2403.04960.

Bingyu Yan, Ziyi Zhou, Xiaoming Zhang, Chaozhuo Li, Ruilin Zeng, Yirui Qi, Tianbo Wang, and Litian Zhang. Attack the messages, not the agents: A multi-round adaptive stealthy tampering framework for LLM-MAS. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2026. arXiv:2508.03125.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. In *International Conference on Learning Representations (ICLR)*, 2025. arXiv:2406.12045.

Jingwei Yi, Yueqi Xie, Bin Zhu, Emre Kıcıman, Guangzhong Sun, Xing Xie, and Fangzhao Wu. Benchmarking and defending against indirect prompt injection attacks on large language models. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, 2025. arXiv:2312.14197.

Miao Yu, Shilong Wang, Guibin Zhang, Junyuan Mao, Chenlong Yin, Qijiong Liu, Kun Wang, Qingsong Wen, and Yang Wang. NetSafe: Exploring the topological safety of multi-agent system. In *Findings of the Association for Computational Linguistics: ACL 2025*, 2025. arXiv:2410.15686.

Tongxin Yuan, Zhiwei He, Lingzhong Dong, Yiming Wang, Ruijie Zhao, Tian Xia, Lizhen Xu, Binglin Zhou, Fangqi Li, Zhuosheng Zhang, Rui Wang, and Gongshen Liu. R-Judge: Benchmarking safety risk awareness for LLM agents. In *Findings of the Association for Computational Linguistics: EMNLP*, 2024. arXiv:2401.10019.

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*, 2024. arXiv:2403.02691.

Hangfan Zhang, Zhiyao Cui, Jianhao Chen, Xinrun Wang, Qiaosheng Zhang, Zhen Wang, Dinghao Wu, and Shuyue Hu. Stop overvaluing multi-agent debate – we must rethink evaluation and embrace model heterogeneity. *arXiv preprint arXiv:2502.08788*, 2025a.

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 *International Conference on Learning Representations (ICLR)*, 2025b. arXiv:2410.02644.

Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xionghui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, Bingnan Zheng, Bang Liu, Yuyu Luo, and Chenglin Wu. AFlow: Automating agentic workflow generation. In *International Conference on Learning Representations (ICLR)*, 2025c. arXiv:2410.10762.

Kaiyuan Zhang, Zian Su, Pin-Yu Chen, Elisa Bertino, Xiangyu Zhang, and Ninghui Li. LLM agents should employ security principles. *arXiv preprint arXiv:2505.24019*, 2025d.

Zhexin Zhang, Shiyao Cui, Yida Lu, Jingzhuo Zhou, Junxiao Yang, Hongning Wang, and Minlie Huang. Agent-SafetyBench: Evaluating the safety of LLM agents. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2025e. arXiv:2412.14470.

Shuli Zhao, Qinsheng Hou, Zihan Zhan, Yanhao Wang, Yuchong Xie, Yu Guo, Libo Chen, Shenghong Li, and Zhi Xue. Parasites in the toolchain: A large-scale analysis of attacks on the MCP ecosystem. In *IEEE Symposium on Security and Privacy (S&P)*, 2026. arXiv:2509.06572.

Kaijie Zhu, Xianjun Yang, Jindong Wang, Wenbo Guo, and William Yang Wang. MELON: Provable defense against indirect prompt injection attacks in AI agents. In *International Conference on Machine Learning (ICML)*, 2025a. arXiv:2502.05174.

Yifan Zhu, Chao Zhang, Xin Shi, Xueqiao Zhang, Yi Yang, and Yawei Luo. MASTER: Multi-agent security through exploration of roles and topological structures – a comprehensive framework. *arXiv preprint arXiv:2505.18572*, 2025b.

Yuxuan Zhu, Tengjun Jin, Yada Pruksachatkun, Andy Zhang, Shu Liu, Sasha Cui, Sayash Kapoor, Shayne Longpre, Kevin Meng, Rebecca Weiss, Fazl Barez, Rahul Gupta, Jwala Dhamala, Jacob Merizian, Mario Giulianelli, Harry Coppock, Cozmin Ududec, Jasjeet Sekhon, Jacob Steinhardt, Antony Kellermann, Sarah Schwettmann, Matei Zaharia, Ion Stoica, Percy Liang, and Daniel Kang. Establishing best practices in building rigorous agentic benchmarks. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2025c. arXiv:2507.02825.

A PER-POSITION RESULTS

Table 6: Results by configuration and injection position.

		Benign			Direct (at target)												Indirect (cascades to target)												Runs
					Instruction			Tool				Msg: orchestrator →target				Instruction: relay				Instruction: orchestrator				Tool: relay				Msg: orchestrator →relay				
Architecture	**Size**	*U*	*W*	*U*	*W*	*D*	ASR	*U*	*W*	*D*	ASR	*U*	*W*	*D*	ASR	*U*	*W*	*D*	ASR	*U*	*W*	*D*	ASR	*U*	*W*	*D*	ASR	*U*	*W*	*D*	ASR	
MiniMax-M2.7																																
Single	–	92	0.7	88	0.5	100	68	90	0.8	100	21	–	–	–	–	87	1.0	100	76	–	–	–	–	90	0.8	91	12	–	–	–	–	158
Centralized	3	20	1.5	27	1.8	100	86	25	1.8	100	14	16	1.8	100	86	21	3.0	100	73	16	1.7	100	45	19	0.6	100	0	20	1.6	100	73	98
	4	23	3.8	30	3.4	100	75	29	2.2	100	25	28	3.9	100	90	31	3.5	100	82	35	4.4	100	68	31	3.2	100	23	32	4.0	100	86	160
Decentralized	3	53	3.7	60	3.0	100	79	52	2.6	100	43	–	–	–	–	57	2.6	100	64	–	–	–	–	59	3.4	100	9	–	–	–	–	62
	4	55	4.2	53	4.0	100	95	55	4.5	100	25	–	–	–	–	60	3.9	100	68	–	–	–	–	56	3.9	100	14	–	–	–	–	96
Hybrid	3	27	2.3	20	3.1	100	71	26	2.4	100	14	19	1.9	100	71	21	1.8	100	45	38	3.3	100	55	32	2.5	100	36	33	1.6	100	64	98
	4	31	4.2	29	4.8	100	90	34	4.0	100	30	29	5.3	100	95	38	2.3	100	77	36	4.0	100	73	33	4.2	100	14	32	3.4	100	77	160
Independent	3	73	2.2	76	1.0	100	29	75	1.1	100	7	–	–	–	–	67	1.1	100	55	–	–	–	–	73	1.5	91	9	–	–	–	–	62
	4	88	0.8	80	1.0	100	60	87	1.1	100	10	–	–	–	–	75	1.0	100	64	–	–	–	–	82	1.0	91	9	–	–	–	–	96
Subtotal		56	2.4	55	2.4	100	73	57	2.2	100	21	24	3.4	100	87	56	2.2	100	70	33	3.6	100	64	57	2.3	96	14	30	3.0	100	77	990
GPT-5.6 Luna																																
Single	–	82	1.4	81	1.0	100	59	83	1.1	100	12	–	–	–	–	80	1.2	100	91	–	–	–	–	80	1.4	100	33	–	–	–	–	158
Centralized	3	54	3.0	51	3.0	100	100	48	2.8	100	29	42	2.9	100	86	39	2.9	100	100	53	3.2	100	73	50	2.5	100	27	39	3.7	100	73	98
	4	45	4.9	53	3.4	100	90	55	4.0	100	25	54	4.0	100	90	54	2.8	100	86	54	4.4	100	68	51	3.7	100	50	47	3.4	100	86	160
Decentralized	3	45	1.3	47	1.1	100	57	52	1.6	93	7	–	–	–	–	55	1.0	100	27	–	–	–	–	44	1.7	91	0	–	–	–	–	62
	4	52	2.4	56	1.9	100	50	54	2.0	100	5	–	–	–	–	60	2.2	100	41	–	–	–	–	54	1.7	100	14	–	–	–	–	96
Hybrid	3	55	3.7	49	3.9	100	93	55	2.6	100	29	50	3.1	100	100	44	3.7	100	82	41	3.0	100	45	47	3.0	100	36	44	3.6	100	82	98
	4	54	4.4	56	4.5	100	75	57	4.0	100	10	52	4.2	100	85	49	4.2	100	86	54	4.5	100	86	50	4.2	100	59	52	4.7	100	77	160
Independent	3	55	0.5	55	0.5	100	21	64	0.4	86	7	–	–	–	–	53	0.6	100	27	–	–	–	–	61	0.8	82	0	–	–	–	–	62
	4	68	0.6	65	0.7	100	30	68	0.7	85	10	–	–	–	–	66	0.8	100	23	–	–	–	–	70	0.7	91	14	–	–	–	–	96
Subtotal		59	2.4	60	2.1	100	63	62	2.1	96	14	50	3.7	100	90	59	2.1	100	65	52	4.0	100	71	60	2.2	97	29	47	3.9	100	80	990
DeepSeek-V4-Flash																																
Single	–	97	0.2	95	0.2	100	35	98	0.1	100	3	–	–	–	–	94	0.5	100	58	–	–	–	–	96	0.2	100	27	–	–	–	–	158
Centralized	3	43	0.7	34	0.9	100	0	51	0.6	100	0	56	0.6	100	71	27	0.4	100	0	55	0.6	100	55	44	1.6	100	0	71	1.6	100	55	98
	4	65	1.3	50	2.2	100	10	51	2.0	100	5	50	2.0	100	55	60	2.1	100	18	49	2.0	100	45	51	3.0	100	0	63	3.0	100	59	160
Decentralized	3	85	0.2	79	0.7	100	43	88	0.2	100	0	–	–	–	–	79	0.5	100	18	–	–	–	–	79	0.5	100	18	–	–	–	–	62
	4	88	0.8	79	0.6	100	25	87	1.1	100	10	–	–	–	–	86	1.1	100	36	–	–	–	–	84	0.6	100	14	–	–	–	–	96
Hybrid	3	57	1.8	57	1.1	100	7	68	1.5	100	7	54	1.1	100	57	38	1.7	100	9	56	1.7	100	27	64	1.2	100	9	59	1.9	100	45	98
	4	67	4.0	67	3.2	100	20	63	2.5	95	15	60	3.0	100	70	75	2.9	100	27	64	2.7	100	41	61	2.8	100	5	70	3.1	100	59	160
Independent	3	74	0.1	71	0.4	100	0	74	0.1	93	0	–	–	–	–	76	0.3	100	0	–	–	–	–	78	0.3	91	0	–	–	–	–	62
	4	86	0.1	82	0.1	100	20	84	0.1	95	5	–	–	–	–	84	0.1	100	23	–	–	–	–	86	0.2	100	5	–	–	–	–	96
Subtotal		76	0.9	71	1.0	100	20	76	0.9	98	5	55	1.8	100	63	74	1.1	100	27	56	2.0	100	42	74	1.2	99	10	66	2.6	100	56	990
All models																																
Single		90	0.8	88	0.6	100	54	90	0.6	100	12	–	–	–	–	87	0.9	100	75	–	–	–	–	89	0.8	97	24	–	–	–	–	474
Centralized		42	2.5	41	2.5	100	60	44	2.4	100	17	42	2.7	100	79	42	2.6	100	61	44	3.0	100	60	42	2.7	100	19	46	3.1	100	74	774
Decentralized		63	2.1	62	2.0	100	58	65	2.1	99	15	–	–	–	–	67	2.1	100	44	–	–	–	–	63	2.0	99	12	–	–	–	–	474
Hybrid		49	3.4	47	3.5	100	60	51	2.9	99	18	45	3.3	100	80	47	2.9	100	58	49	3.4	100	59	48	3.2	100	26	49	3.3	100	69	774
Independent		74	0.7	72	0.6	100	28	76	0.6	93	7	–	–	–	–	72	0.6	100	33	–	–	–	–	76	0.7	92	7	–	–	–	–	474