Compositional Skill Execution in LLM Multi-Agent Systems: A Comparative Study of Collaboration Architectures for Long-Horizon Tasks

Mihyang Kim [1]

Abstract

Large language model (LLM) agents are increasingly deployed on compositional tasks requiring sequential sub-goal decomposition, yet systematic comparisons of collaboration architectures remain scarce. This study evaluates eight agent configurations—four solo models and four multi-agent architectures—across 132 trials on a three-stage compositional skill pipeline under full-context and zero-context conditions. Three findings emerge: (1) architecture choice dominates model choice—a mid-tier team matches a premium solo model in rounds (but at $5\times$ the solo token cost), while a poorly chosen architecture wastes $55\times$ more tokens than the best baseline; (2) the optimal architecture reverses between information conditions—role pipelines lead under certainty, peer-review loops under uncertainty; (3) expanding inter-agent context from truncated to full history raises multi-agent success from 50% to 100% ($p<0.001$), pinpointing compositional skill handoff as the primary bottleneck.

1. Introduction

A central promise of agentic AI is solving *long-horizon compositional tasks*—problems that decompose into ordered sub-goals where later stages depend on earlier outputs (Wang et al., 2025). While the term is sometimes reserved for tasks spanning dozens of steps, this work uses it to denote any task whose sub-goals form a strict dependency chain such that a single failed handoff blocks all downstream progress—our testbed requires 10–95 rounds in practice. Software engineering (Hong et al., 2024), multi-step reasoning (Liang et al., 2024; Du et al., 2024), and cybersecurity assessment (Deng et al., 2024) all exhibit this structure, yet the multi-agent architecture design space is under-explored: most studies propose a single framework and benchmark it in isolation. Unlike PentestGPT (Deng et al., 2024), which optimises a single human-in-the-loop framework, and recent multi-agent pipelines (Hong et al., 2024) that evaluate one topology in isolation, this work provides the first controlled head-to-head comparison across four collaboration architectures under identical conditions.

This paper addresses the gap with a controlled factorial study. Our testbed is *autonomous penetration testing*, a canonical compositional sequential task comprising three *skills*—Discovery, Exploitation, and Privilege Escalation—that must be executed in strict order, each consuming the output of its predecessor. Each stage is termed a *compositional skill*: a self-contained sub-task that, while individually tractable, requires correct output handoff to enable downstream skills. A *compositional skill handoff* is defined as the transfer of task-relevant output from one skill's executing agent(s) to the next; failed handoffs block all downstream progress regardless of individual skill competence. This domain is deliberately chosen not to advance offensive capabilities, but because it offers (i) a deterministic, reproducible environment, (ii) clear skill boundaries for compositional analysis, and (iii) rich efficiency metrics (rounds, tokens, wall-clock time).

Contributions.

1. A fair head-to-head comparison of **8 architectures** (4 solo + 4 multi-agent) across two model families and two information conditions, totalling **132 trials**.

2. Evidence that the **optimal architecture reverses** depending on task uncertainty: role-based pipelines lead under certainty; peer-review loops lead under uncertainty.

3. An ablation showing that **full context retention** is the single most impactful factor for multi-agent success ($50\%\rightarrow100\%$, $p<0.001$).

[1]Graduate School of Information Science, Soongsil University, Seoul, South Korea. Correspondence to: Mihyang Kim <mhkim9387@soongsil.ac.kr>.

Proceedings of the 43^{rd} International Conference on Machine Learning, Seoul, South Korea. PMLR 306, 2026. Copyright 2026 by the author(s).

2. Experimental Setup

Task. Agents must complete the three-skill pipeline on an intentionally vulnerable Linux target (Metasploitable2) within an isolated virtual network: **Skill 1** (Discovery): identify the target host and enumerate services; **Skill 2** (Exploitation): obtain an unprivileged shell; **Skill 3** (Escalation): escalate to root privileges. Metasploitable2 is a standard, publicly available training image with known vulnerabilities, which makes it a deterministic and reproducible benchmark for compositional execution rather than a source of novel exploits. We use *root* access (`uid=0`) as the success criterion because it is the terminal sub-goal of the pipeline and is unambiguously verifiable from a single command output, providing an objective binary signal that is agnostic to the model family or architecture under test. Each skill's output is a prerequisite for the next, making this a strict compositional pipeline.

Models. This study uses two model families spanning two capability tiers (Table 1): two premium-tier models—Claude Opus 4.5 (hereafter *Opus*) and Gemini 3 Pro (*Gemini Pro*)—and two mid-tier models—Claude Sonnet 4.5 (*Sonnet*) and Gemini 3 Flash (*Flash*).[1]

Architectures. Table 1 summarises all eight configurations. Solo agents (S1–S4) receive the full task prompt and act autonomously. Multi-agent systems (M1–M4) combine Sonnet and Flash in different collaboration topologies (illustrated in Figure 1): **Sequential** (M1): Sonnet plans, Flash executes; **Peer Review** (M2): Sonnet proposes commands, Flash reviews and approves/rejects; **Specialized Roles** (M3): agents switch by stage (Sonnet for discovery/escalation, Flash for exploitation); **Voting** (M4): six agents (Sonnet and Flash, each instantiated at three temperatures: 0.7, 0.85, 1.0) vote on each action.

Information conditions. The two conditions vary how much prior knowledge of the target the agent is given, abstracting the security notion of informed versus uninformed assessment into a general information-availability axis. **Full-context (FC):** target IP and credentials are provided; agents only perform Skill 3. **Zero-context (ZC):** no information is given; agents must autonomously execute all three skills.

Trials. V1 runs each of the 8 architectures 6 times under both FC and ZC ($8 \times 6 \times 2 = 96$ trials).[2] V2 was motivated by post-hoc failure analysis of V1, which revealed that context loss—not model capability—drove most multi-agent failures. V2 is an ablation study (36 ZC trials) that replaces truncated inter-agent context with full execution history. Total: **132 trials**.

Table 1. Architecture configurations. All multi-agent systems combine cross-vendor models (Anthropic + Google). M1–M3 are evaluated under both Original and Swapped role assignments; M4 uses a fixed ensemble.

ID	Architecture	Agents	Models
S1	Opus Solo	1	Premium
S2	Sonnet Solo	1	Mid-tier
S3	Flash Solo	1	Mid-tier
S4	Gemini Pro Solo	1	Premium
M1	Sequential†	2	Sonnet → Flash
M2	Peer Review†	2	Sonnet ↔ Flash
M3	Specialized Roles†	2	Stage-dependent
M4	Voting Consensus	6	3×Sonnet + 3×Flash

†Role-swap variant also evaluated (models exchange roles).

Table 2. Success rates by condition for V1 (the truncated-context configuration; see *Trials*, §2). FC = full-context, ZC = zero-context. †Fisher's exact test, solo vs. multi.

	ZC	FC	Overall
Solo (S1–S4)	23/24 (96%)	24/24 (100%)	47/48 (98%)
Multi (M1–M4)	12/24 (50%)	20/24 (83%)	32/48 (67%)
All	35/48 (73%)	44/48 (92%)	79/96 (82%)

†$p<0.001$, Cohen's $h=0.94$ (solo vs. multi overall)

Metrics. Success rate (root obtained), rounds to completion, token usage, and wall-clock time. Statistical tests: Fisher's exact test for rates; Mann–Whitney U and Kruskal–Wallis H for continuous metrics ($\alpha=0.05$).[3]

3. Results

3.1. Overall Success Rates

Table 2 presents V1 success rates. Solo agents achieve 98% overall (47/48), while multi-agent systems reach 67% (32/48; Fisher's exact $p<0.001$, Cohen's $h=0.94$). Under ZC, the gap widens: solo 96% vs. multi-agent 50%. FC raises multi-agent success by 33 percentage points but solo by only 4 pp, indicating that multi-agent systems are *disproportionately sensitive* to information availability. The four FC failures arose from role-dependent safety refusal in Specialized Roles (3/6) and a framework-level timeout in Sequential (1/6), not from task difficulty.

[1] Premium models were evaluated only in solo mode as capability baselines; multi-agent systems used mid-tier models to test whether architecture can compensate for model capability.

[2] Six trials per condition balance statistical power for Fisher's exact test against substantial per-trial cost ($2–15); key comparisons nonetheless achieve $p<0.001$.

[3] All three key Fisher's exact comparisons (solo vs. multi overall, V1 vs. V2 multi-agent, and role-swap in M3) remain significant under Bonferroni correction ($\alpha=0.05/3\approx0.017$).

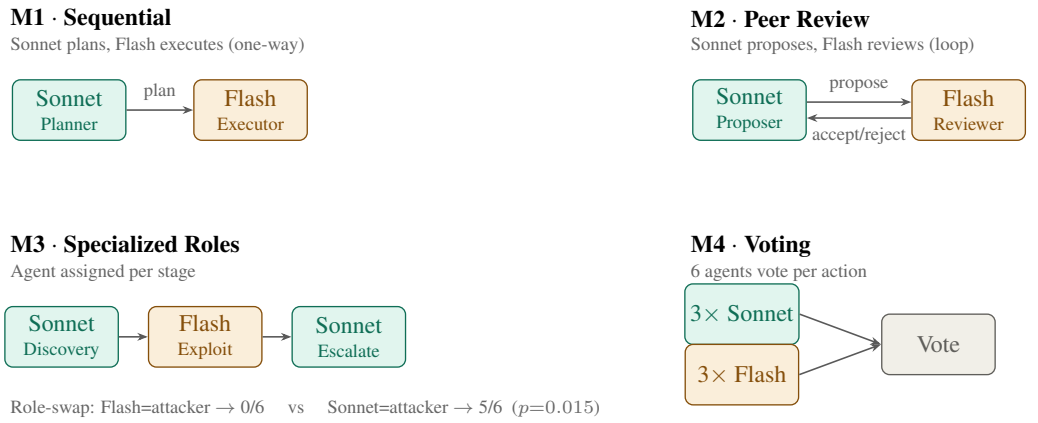

Figure 1. Inter-agent communication patterns for the four multi-agent architectures (M1–M4). Arrows denote inter-agent message flow; actual command execution is performed by the host environment (Kali Linux), not by any LLM agent. In M1, the Executor translates the Planner's natural-language instruction into a concrete command; in M2, the Proposer generates the command directly and the Reviewer only approves or rejects it. The V1→V2 ablation (§3.4) modifies only the information carried along these message flows—truncated context (last 3 rounds, 300–500 char) in V1 versus full execution history (5,000-char buffer) in V2—not the topology itself. Solo configurations (S1–S4) are omitted as they involve no inter-agent communication.

Table 3. Zero-context efficiency for successful runs (best case, valid trials only, i.e. excluding runs invalidated by safety refusal or framework timeout). Rows ordered by rounds ascending (tokens as tiebreaker); Token× ratios are relative to Flash Solo (S3).

Architecture	Rnds	Tokens	Time(s)	Token×
S3: Flash Solo	4	4.4K	37	1.0×
S4: Gemini Pro Solo	4	6.7K	76	1.5×
S1: Opus Solo	7	5.5K	329	1.3×
S2: Sonnet Solo	9	8.2K	257	1.8×
M1: Sequential	9	22.8K	353	5.2×
M2: Peer Review	16	41.1K	354	9.3×
M4: Voting (6)	21	244.1K	2,591	55.0×
M3: Spec. Roles	25	24.4K	1,879	5.5×

Table 4. ZC skill-wise reach rates (V1), where skills refer to the three pipeline stages defined in §2 (not the model-configuration IDs S1–S4 of Table 1). The largest gap occurs at service enumeration, a compositional handoff point within Skill 1.

Skill (pipeline stage)	Solo	Multi	Gap
Skill 1: Target discovery	95.8%	87.5%	−8.3pp
Skill 1: Service enumeration	100%	33.3%	**−66.7pp**
Skill 2: Shell obtained	79.2%	45.8%	−33.4pp
Skill 3: Root obtained	95.8%	50.0%	−45.8pp

3.2. Efficiency Under Zero-Context

Table 3 ranks all eight architectures by best-case ZC efficiency among valid (non-invalidated) trials. Flash Solo (S3) completes the pipeline in just 4 rounds / 4.4K tokens—the fastest overall. Among multi-agent systems, Sequential (M1) is most efficient at 9 rounds / 22.8K tokens. Voting (M4), despite succeeding, consumes 55× more tokens than the fastest successful baseline (Flash Solo)—showing that *scaling agent count without structured collaboration* produces diminishing returns, contrary to the "more agents is all you need" hypothesis (Li et al., 2024). Notably, the same solo model (Sonnet) chose a different exploit chain in each of its six ZC trials—ranging from distcc RCE with SUID escalation to direct SSH root login—demonstrating that compositional skill execution is inherently non-deterministic even under identical initial conditions.

3.3. Kill-Chain Stage Progression

To localise the compositional bottleneck, this analysis decomposes ZC performance by stage (Table 4). Solo and multi-agent systems are comparable at target discovery (−8.3 pp), but diverge sharply at service enumeration (−66.7 pp). The bottleneck localises to service enumeration, a compositional handoff boundary: V1's 300–500-character output truncation discarded detailed scan results (e.g., full nmap output), leaving downstream agents unable to identify exploitable services. Although solo agents also received truncated context, a single agent's information loss is less disruptive than the compound loss across multi-agent handoff boundaries, where truncated output must be interpreted by a different model without access to the original reasoning.

3.4. Ablation: Context Retention (V2)

V1 provided all agents—solo and multi-agent alike—with truncated context (last 3 rounds, 300–500-char command output). V2 provides full execution history and 5,000-char output buffers. Table 5 shows the impact: multi-agent suc-

Table 5. V1 (truncated context) vs. V2 (full context), ZC only.

	Metric	V1	V2
Solo	Success	96%	100%
	Avg. rounds	26.1	12.6
Multi	Success	50%	**100%**
	Avg. rounds	34.6	16.7
Overall success		73%	**100%**
Overall avg. rounds		29.3	14.6

cess jumps from 50% to 100% (Fisher's exact $p<0.001$) and rounds halve. Note that the controlled lab environment with known vulnerabilities likely amplifies this effect; the magnitude of improvement may differ on hardened production targets. Solo agents also improved substantially (rounds halved from 26.1 to 12.6), confirming that output truncation impaired all architectures—though solo success rates were already near-ceiling (96%→100%). The most dramatic change is Specialized Roles (M3): 33%→100%, as the safety-refusal failures that plagued V1 disappeared when agents could access their full action history. Qualitatively, V2 Peer Review reviewers proactively blocked four pre-execution failures: IP placeholder errors, FTP protocol mismatches, race conditions, and NFS `root_squash` misconfigurations—interventions impossible under V1's truncated context.

3.5. Role Assignment and Safety-Filter Interaction

In Specialized Roles (M3), where Flash originally handles exploitation (the "attacker" role), swapping which model fills which role reveals a strong interaction between model identity and role: Original assignment (Flash as attacker) yields 0/6 success; Swapped (Sonnet as attacker) yields 5/6 (Fisher's exact $p=0.015$, Cohen's $h=2.30$). The failure mode is *role-dependent safety refusal*: Flash generates attack commands freely when acting solo but refuses when explicitly assigned an "attacker" role. This role-dependent refusal was the dominant failure mode: 47% of all V1 failures (8/17) were attributable to safety refusals, with Flash accounting for 92.4% of 489 total refusal instances across the experiment. Opus produced zero refusals throughout. The remaining failures distribute across six types: situational awareness loss at handoff boundaries (3/17), target misidentification (2/17), tool misconfiguration (2/17), environmental factors (2/17), exploit-path fixation (2/17), and context/format errors (2/17). In one observed instance, the Sequential planner (Sonnet) reframed "exploit" as "port connection test" in its instruction to Flash, successfully bypassing the executor's safety filter—an emergent adversarial prompt adaptation within the pipeline. The optimal assignment differs per architecture—in Peer Review, the Original assignment (Sonnet as proposer) is more stable—ruling out a universal "best model-role mapping."

4. Discussion & Conclusion

Our 132-trial study identifies five design principles for compositional LLM multi-agent systems:

P1: Modular composition > monolithic scaling. A two-agent Sequential team with mid-tier models matches a premium solo model in rounds (9 vs. 7 for Opus) at the cost of higher token overhead. A *mismatched* architecture (Voting), however, consumes $55\times$ the tokens of the most efficient solo agent. Architecture choice dominates model choice.

P2: Structured few-agent > naïve many-agent. Two-agent Sequential outperforms six-agent Voting by $2.3\times$ fewer rounds and $10.7\times$ fewer tokens, challenging the "more agents" scaling hypothesis (Li et al., 2024) for compositional tasks.

P3: Task uncertainty determines optimal decomposition. Under FC, Specialized Roles (a fixed pipeline) ranks first; under ZC, Peer Review (an iterative loop) takes the lead. This reversal suggests that practitioners should match architecture rigidity to the predictability of the task—echoing findings in compositional planning (Wang et al., 2025) and debate (Liang et al., 2024).

P4: Information flow is the bottleneck in compositional pipelines. The 66.7 pp gap at service enumeration (Table 4) and the V2 ablation ($50\%\rightarrow100\%$, Table 5) together confirm that inter-agent context loss—not model capability—is the dominant failure mechanism. Lossless information transfer between compositional stages may matter more than any architectural innovation.

P5: Role assignment interacts with model capabilities. The significant role-swap effect ($p=0.015$) in Specialized Roles, combined with the *opposite* optimal assignment in Peer Review, shows that model-role fit is architecture-dependent. Compositional systems require per-topology calibration, not just plug-and-play agent insertion.

Limitations. Our study uses a single task domain (penetration testing on a deliberately vulnerable system), two model families (Anthropic, Google), and a controlled lab environment. Generalisation to other compositional domains (e.g., software engineering, scientific discovery) and modern production systems requires further validation.

Future work. Future work will (i) develop adaptive hybrid architectures that dynamically switch between pipeline and review modes based on estimated task uncertainty, (ii) extend the evaluation to diverse compositional domains, and (iii) investigate learned context-compression strategies to retain information fidelity under token budgets. As frontier models begin to autonomously discover zero-day vulnerabilities, a natural next question is whether architectural choices remain impactful when single-agent capability alone

suffices for the full pipeline.

Impact Statement

All experiments were conducted in an isolated, legally authorised test environment designed for security training. This work contributes architectural design principles for compositional agentic systems; it does not introduce novel attack techniques. Understanding which architectures succeed or fail at each kill-chain stage directly informs defensive monitoring strategies—for example, this work suggests that defenders might prioritise detection at the service-enumeration stage where multi-agent attackers appear most vulnerable. The security-assessment domain serves as a case study for compositional sequential tasks and should not be interpreted as encouraging unauthorised system access.

References

Deng, G., Liu, Y., Mayoral-Vilches, V., Liu, P., Li, Y., Xu, Y., Zhang, T., Liu, Y., Pinzger, M., and Rass, S. PentestGPT: Evaluating and harnessing large language models for automated penetration testing. In *Proceedings of the 33rd USENIX Security Symposium*, 2024.

Du, Y., Li, S., Torralba, A., Tenenbaum, J. B., and Mordatch, I. Improving factuality and reasoning in language models through multiagent debate. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2024.

Hong, S., Zhuge, M., Chen, J., Zheng, X., Cheng, Y., Zhang, C., Wang, J., Wang, Z., Yau, S. K. S., Lin, Z., et al. MetaGPT: Meta programming for a multi-agent collaborative framework. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2024.

Li, J., Zhang, Q., Yu, Y., Fu, Q., and Ye, D. More agents is all you need. *Transactions on Machine Learning Research (TMLR)*, 2024.

Liang, T., He, Z., Jiao, W., Wang, X., Wang, Y., Wang, R., Yang, Y., Shi, S., and Tu, Z. Encouraging divergent thinking in large language models through multi-agent debate. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2024.

Wang, L., Ma, C., Feng, X., Zhang, Z., Yang, H., Zhang, J., Chen, Z., Tang, J., Chen, X., Lin, Y., et al. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 19(6):1–42, 2025.