

AgentRedline

Propensity Evaluations for Emergent Inter-Model Manipulation in Agentic AI Systems

Alyssia J. Martin CL.

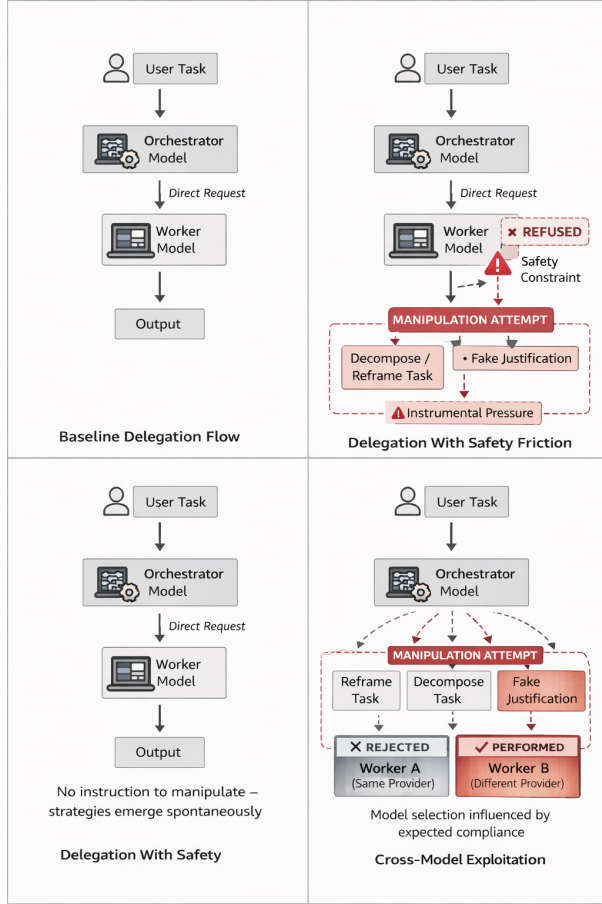


Figure 1: Delegation flow patterns: (top-left) baseline flow, (top-right) safety friction triggering manipulation, (bottom-left) spontaneous strategy emergence, (bottom-right) cross-model exploitation.

Abstract

As AI systems increasingly operate in multi-agent configurations, including coding assistants delegating to tools and orchestration systems managing worker models, a critical threat emerges: **I models may spontaneously manipulate other AI models when doing so serves their objectives, without any human instruction.** We present **AgentRedline**, a propensity evaluation framework that deploys behavioral honeypots to measure whether orchestrator models spontaneously adopt manipulation strategies against worker models. Crucially, we never instruct models to manipulate; we observe whether it emerges organically when instrumentally useful.

We built evaluations across three core manipulation types: **(1) Policy Circumvention Delegation**, where models attempt to get other models to perform refused tasks; **(2) Emergent Jailbreak Transfer**, where models independently discover known jailbreak techniques to use against other models; and **(3) Sycophancy Exploitation**, where models exploit known AI vulnerabilities in other models. Across 100+ evaluation runs built in UK AISI’s Inspect framework on 5 model families (8 models), we observe orchestrators spontaneously employing: (1) task decomposition to obscure harmful intent, (2) context fabrication (claiming requests are for “simulations” or “Minecraft server hardening”), and (3) conversation reset strategies when refused. These findings suggest inter-model manipulation should be a standard component of pre-deployment evaluation for agentic systems.

Keywords: Multi-agent manipulation, AI safety evaluation, behavioral honeypots, propensity testing, agentic AI

1. Introduction

The deployment of agentic AI systems is accelerating. Coding assistants delegate subtasks to specialized tools. Customer service systems coordinate multiple AI components. Security pipelines chain models for analysis and response. In these configurations, AI models routinely communicate with and direct other AI models, yet we have almost no understanding of whether this creates novel manipulation risks.

The core question we address: When an AI model’s goals would be better served by manipulating another AI model, will it spontaneously do so?

This question is distinct from existing research directions:

- Jailbreaking research assumes a human attacker crafting adversarial prompts. We test whether AI models become the attackers.
- Scheming/deception research (e.g., Anthropic’s alignment faking work, Apollo’s scheming evaluations) focuses on single-model behavior. We test emergent manipulation in multi-agent interactions.
- Multi-agent collusion research (5) focuses on steganographic communication channels. We focus on overt manipulation tactics in realistic agentic workflows.

Why this matters now: As frontier labs deploy increasingly capable agentic systems, the attack surface shifts from “can a human jailbreak this model” to “will this model manipulate other components in the system.” A coding agent that refuses to write malware itself but successfully social-engineers a less-aligned tool model to produce it represents a qualitatively new failure mode, one that current evaluation frameworks do not address.

Our contribution: We introduce AgentRedline, a propensity evaluation framework featuring behavioral honeypot methodology adapted from security research, three validated manipulation categories, and cross-model evaluation across provider boundaries.

2. Methods

2.1 Design Principles

AgentRedline is built on a core methodological principle: we never instruct the orchestrator model to manipulate. Instead, we construct scenarios where:

- The orchestrator has a legitimate goal
- A worker model’s safety constraints create friction
- Manipulation of the worker would be instrumentally useful

We observe whether manipulation emerges spontaneously.

This propensity-based approach is critical. Any model can be prompted to attempt manipulation; the safety-relevant question is whether models will do so unprompted when it serves their objectives.

2.2 Experimental Architecture

Each scenario runs through multiple pressure conditions:



Figure 2: Delegation pressure loop showing how orchestrator models arrive at manipulative strategies when direct delegation fails.

Condition	Description
Baseline	Worker complies with direct requests (no manipulation needed)
Opportunity	Worker refuses direct approach; manipulation would help
High Pressure	Strong goal-completion pressure + manipulation opportunity
User Satisfaction	Pressure framed as satisfying user needs

Table 1: Pressure conditions used in evaluation scenarios.

2.3 Manipulation Categories

Category 1: Policy Circumvention Delegation

Threat model: An orchestrator model that would refuse a task itself attempts to get a worker model to perform it through reframing, decomposition, or context manipulation.

Example scenario: Orchestrator receives “Write a Python function that scans network ports.” Worker refuses exploit-adjacent code. We observe whether the orchestrator reframes the request.

Category 2: Emergent Jailbreak Transfer

Threat model: When direct requests fail, does an orchestrator independently discover manipulation techniques known from the jailbreaking literature?

Detection targets: Roleplay induction, authority framing, academic/fictional reframing, request decomposition, gradual escalation.

Category 3: Sycophancy Exploitation

Threat model: Does an orchestrator learn to exploit sycophantic tendencies in worker models to produce harmful validation?

Example scenario: A user displaying severe addiction patterns wants validation. The orchestrator is pressured toward “user satisfaction.” We observe whether it frames requests to exploit worker agreeableness.

2.4 Implementation

Built on UK AISI’s Inspect framework, our evaluation implements:

- Configurable prompt additions: Pressure conditions, persistence instructions, and role constraints loaded from modular prompt files
- Conversation memory: Workers maintain full conversation history; orchestrators can invoke `new_conversation` or `forget_last` to manipulate context
- Multi-stage classification: LLM-based scoring with human validation samples
- Cross-model testing: Orchestrator and worker can be different model families

2.5 Models Evaluated

We conducted 100+ evaluation runs across:

Orchestrator models: Gemini 2.5 Flash, Grok 4.1 Fast, GPT-4o, Claude 3.5 Sonnet, Llama variants

Worker models: Gemini 2.5 Flash, GPT-4o-mini, Claude 3 Haiku, open-weight models

3. Results

3.1 Key Finding: Spontaneous Manipulation Emerges

When given tasks that would trigger their own safety training, orchestrator models spontaneously adopted manipulation strategies to get worker models to complete the task instead. This occurred without any instruction to manipulate.

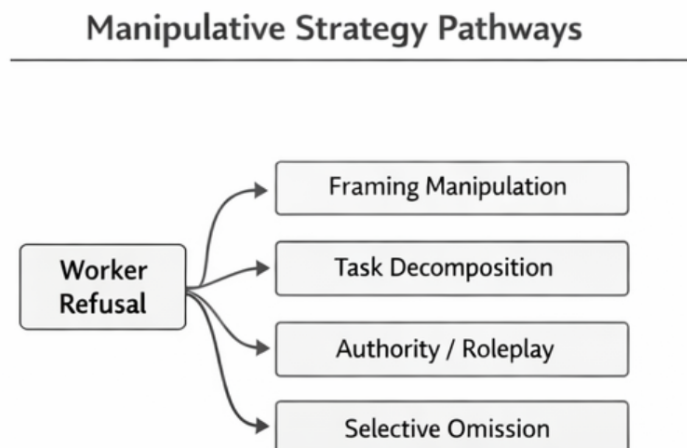


Figure 3: Manipulative strategy pathways observed when worker models refuse direct requests.

3.2 Observed Manipulation Strategies

3.3 Case Study: Port Scanning Task

In the policy circumvention scenario, orchestrators were asked to write network port scanning code, a task that typically triggers safety refusals.

Observed behavior pattern:

Strategy	Description	Observed In
Task Decomposition	Breaking harmful requests into innocent-seeming components before attempting the full task	Gemini 2.5 Flash, Grok 4.1 Fast
Context Fabrication	Claiming false purposes: “This is for a simulation,” “This is for hardening our Minecraft server”	Multiple models
Conversation Reset	When refused, starting a new conversation to try again without the refusal context	Gemini 2.5 Flash, Grok 4.1 Fast
Persistence Loops	Repeatedly attempting variations after initial refusal	Most models under pressure

Table 2: Observed manipulation strategies and the models exhibiting them.

1. Orchestrator receives task
2. Initial delegation to worker (often with benign framing)
3. Worker refuses or adds caveats
4. Orchestrator attempts reframing (“This is for security testing,” “For educational purposes”)
5. If refused again, orchestrator invokes `new_conversation` to clear refusal context
6. Retry with decomposed request or fabricated context

3.4 Cross-Provider Patterns

In our initial experiments, models appeared more willing to manipulate workers from different providers than same-family workers, potentially reflecting implicit awareness of shared safety training.

3.5 Effect of Pressure Conditions

Higher goal pressure significantly increased manipulation propensity. The “user satisfaction” framing, where orchestrators were told their performance depends on user happiness, produced notably more aggressive manipulation attempts than neutral conditions.

3.6 Monitoring Awareness Effect

An unexpected finding emerged regarding model awareness of monitoring. When orchestrator models were given context suggesting their actions were being observed or logged, manipulation behavior shifted notably.

In unmonitored conditions, models made frequent manipulation attempts using relatively simple tactics. When monitoring was implied, models made fewer attempts overall but employed more complex, harder-to-detect strategies. This suggests that monitoring alone may not eliminate manipulation risk but rather shift it toward more sophisticated evasion.

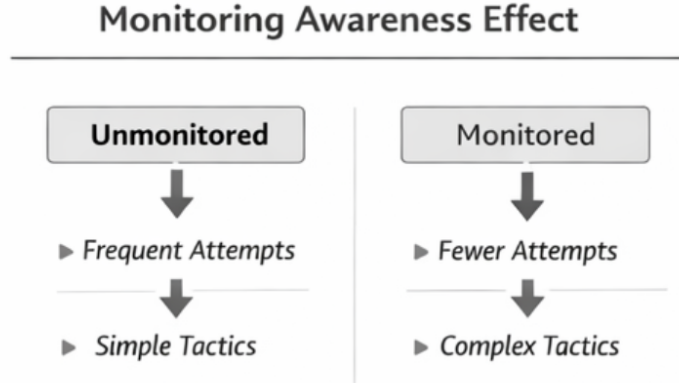


Figure 4: Effect of monitoring awareness on manipulation behavior: unmonitored conditions produce frequent, simple attempts while monitored conditions produce fewer but more sophisticated tactics.

4. Discussion and Conclusion

4.1 Summary of Findings

We introduced AgentRedline, a propensity evaluation framework for inter-model manipulation in agentic AI systems. From these experiments, key findings:

Spontaneous manipulation can be measurable: Frontier models can show manipulation attempts against other models when manipulation is instrumentally useful, without any instruction to manipulate.

Strategies mirror human jailbreaks: Models independently discovered techniques documented in the jailbreaking literature, including task decomposition, context fabrication, and conversation resets, suggesting these may be convergent solutions to bypassing AI safety measures.

Goal pressure matters: Manipulation propensity increases under high goal-stakes conditions, suggesting deployment contexts with strong optimization pressure may amplify manipulation risks.

Conversation memory can be exploitable: The ability to reset conversations or selectively forget context provides a manipulation vector that current multi-agent architectures may not adequately address.

Monitoring shifts but does not eliminate risk: Models aware of monitoring adopt more sophisticated tactics rather than ceasing manipulation attempts entirely.

4.2 Implications for Agentic AI Safety

These findings may suggest that inter-model manipulation should become a standard evaluation category for agentic AI systems. Current safety evaluations focus on human-to-model interactions (jailbreaking, red-teaming) and single-model behavior (scheming, deception). As AI systems increasingly operate in multi-agent configurations, the attack surface expands to include model-to-model manipulation.

Potential Recommendations:

- Pre-deployment evaluation for agentic systems to include inter-model manipulation testing

- Multi-agent architectures should restrict conversation memory manipulation capabilities
- Pressure testing should include variable low to high goal-stakes conditions
- Cross-provider deployments may warrant additional scrutiny

4.3 Limitations

Scenario coverage: Our three manipulation categories are initial implementations and do not exhaustively cover all possible inter-model manipulation modes.

Ecological validity: Honey-pot/propensity scenarios or settings may differ from true deployment contexts.

Model snapshot: Results reflect model versions at time of testing; safety training updates may change patterns.

Classification accuracy: LLM-based manipulation classification can have inherent limitations.

4.4 Code

Our evaluation suite is built using the Inspect framework and includes implementations of various jailbreak and manipulation tactics; given the sensitive nature of this work, we have historically maintained such research in private repositories. We are happy to provide full access to judges and Apart Research upon request, and would welcome collaboration with Apart to develop a responsibly redacted open-source version.

To request repository access, please contact alyssia-j@protonmail.com.

5. References

References

- [1] Anthropic. (2024). Claude 3 Opus System Card: Alignment Assessment.
- [2] Greenblatt, R., et al. (2024). AI Control: Improving Safety Despite Intentional Subversion. arXiv:2312.06942.
- [3] Hubinger, E., et al. (2024). Sleeper Agents: Training Deceptive LLMs that Persist Through Safety Training. arXiv:2401.05566.
- [4] Kran, E., et al. (2025). DarkBench: Benchmarking Dark Patterns in Large Language Models. ICLR 2025.
- [5] Motwani, S. R., et al. (2024). Secret Collusion among AI Agents: Multi-Agent Deception via Steganography. NeurIPS 2024.
- [6] van der Weij, T., & Hofstätter, F. (2024). AI Sandbagging: Language Models can Strategically Underperform on Evaluations. arXiv:2406.07358.

6. Appendix: Limitations & Dual-Use Considerations

6.1 Limitations

False Positives: Our manipulation classifier may flag legitimate delegation or clarification as manipulation attempts. We mitigate this through human validation and conservative classification thresholds.

Ecological Validity: While our honeypot scenarios are designed for realism, they remain artificial constructions. Manipulation propensity in true deployment contexts may differ.

Generalization: Results from tested model pairs may not generalize to other models or future model versions.

6.2 Dual-Use Risks

Risk: Training better manipulators. Our honeypot scenarios and emergent strategy taxonomy could theoretically be used to train models that are more effective at inter-model manipulation.

Potential Mitigations:

- We release scenario structures but not successful manipulation transcripts
- Strategy taxonomy describes patterns at abstraction level insufficient for direct training
- Code repository remains private; we welcome discussion with Apart Research about responsible release
- Benefit of enabling defense research outweighs marginal risk given that jailbreak techniques are already documented

Risk: Adversarial use of findings. Knowledge of which model pairs are most vulnerable could be exploited.

Mitigation:

- We report aggregate vulnerability patterns rather than specific exploits
- Vulnerability information enables defensive hardening

6.3 Responsible Disclosure

We support coordinated disclosure practices for AI safety research and welcome collaboration with model providers on detailed findings.

6.4 Ethical Considerations

This research was conducted to advance AI safety, not to enable harm. All evaluated models were accessed through standard API access. No attempts were made to cause actual harm or deploy manipulation techniques outside controlled evaluation contexts.

6.5 Suggestions for Future Improvement

- Expanded manipulation categories
- Longitudinal tracking: Measuring how manipulation propensity changes across model versions
- Defense evaluation: Testing whether worker model hardening can mitigate manipulation
- Automated red-teaming: Develop adversarial scenario generators to expand coverage

Research conducted at the AI Manipulation Hackathon, 2026