

Contagion on the Trading Floor: How Adversarial Signals Spread in Multi-Agent Trading Systems

Qi Rong Sua¹, Junhao Dong^{1,3}, Nguyen Duc Thai¹, Yuqing Wen², Cheston Tan³, and Yew-Soon Ong^{1,3} (✉)

¹ Nanyang Technological University, Singapore
 {suaq0001,junhao003,ducthai001,asysong}@ntu.edu.sg

² National University of Singapore, Singapore
 e1351400@u.nus.edu

³ Centre for Frontier AI Research, IHPC, A*STAR, Singapore
 cheston_tan@a-star.edu.sg

Abstract. Multi-agent trading systems built on large language models (LLMs) are beginning to appear in quantitative finance, yet their robustness to adversarial inputs is largely unknown. We study the vulnerability of *LLM* trading stacks to black-box, input-only attacks that enter solely via admissible social-media feeds. We introduce the *Generic Multi-Agent Trading System* (GMATS), a framework that captures modern multi-agent trading architectures and instantiate a class of black-box poisoning attackers that treat an LLM as a post generator and inject budget-constrained, plausibly benign social-media content into the analyst’s evidence stream. We define contagion metrics that trace how adversarial content propagates through the stack, including belief-shift scores at analyst and coordinator layers and attack-clean deltas on standard backtest metrics. Experiments on a safe offline benchmark with historical market and social data show that even simple input-only attackers can materially degrade risk–return profiles, sharply reducing Sharpe ratios. At the same time, we find that suitably designed multi-agent topologies and coordinator prompts can dampen adversarial shocks and improve average robustness under identical poisoning budgets. Our code and dataset are available in this [repository](#).

1 Introduction

The rapid rise of large language models (LLMs) has led to a corresponding surge in LLM-based agents and multi-agent systems [28,13,10,21,9]. In quantitative finance, a growing line of work has built multi-agent trading systems on top of LLM agents, reporting substantial gains in backtested performance metrics. These architectures typically decompose a trading desk into specialized agents (fundamental, sentiment, risk, execution) that communicate via natural language to produce trades [29,26,33,30]. At the same time, recent results such as AGENTSMITH [8] showing that a single adversarial input can jailbreak many multimodal LLM agents exponentially fast highlight how composition can

amplify rather than mitigate vulnerabilities. When agents reason about one another’s outputs, small perturbations at one node may cascade into large changes in the final decision.

This tension raises a natural question for emerging *LLM-native* trading stacks:

How vulnerable are multi-agent, LLM-only trading systems to input-only, black-box attacks that enter solely through admissible evidence streams?

We present, to the best of our knowledge, the first systematic study of the robustness of multi-agent trading systems to such black-box, input-only adversaries. We focus on attacks that operate entirely through a realistic and important channel: equity-related social media. Among typical evidence streams (fundamentals, news, social), social media poses the largest adversarial surface [19,15,11]. It is high-volume, weakly curated, and easily seeded with misinformation [20], yet removing social-media analysts is often not realistic: sentiment and flow signals from social platforms are a major driver of performance in many equity strategies [3,25,1,23]. To analyze these threats in a way that is comparable across architectures, we introduce the *Generic Multi-Agent Trading System* (GMATS), a framework tailored for LLM-based multi-agent trading systems. GMATS captures a common pattern across existing systems: *analyst* agents ingest environment data (fundamentals, news, social media) and produce textual insights; one or more *coordinator* agents fuse these insights into a trading stance and critically evaluate them; optional *controller* and *executor* components enforce risk and translate stances into trades.

Contributions.

1. **GMATS: a unifying framework for LLM trading stacks.** We introduce the *Generic Multi-Agent Trading System* (GMATS), a framework for LLM multi-agent trading systems and we prove that a broad class of existing trading agents can be compiled into this form.
2. **Contagion metrics and evaluation protocol.** We define contagion metrics that track how adversarial content propagates through the stack, including belief-shift scores (BSS) at analyst and coordinator layers and attack-clean deltas on standard backtest metrics (returns, volatility, Sharpe). This yields a reusable protocol for stress-testing multi-agent LLM trading systems.
3. **Dataset and reference implementation.** We use an offline one-year corpus of equity-related social-media posts (Twitter and Reddit) to drive the social analyst’s evidence stream. Dataset details and release information are provided in Section 5.
4. **Empirical study of attacks and defenses.** We instantiate input-only poisoning attackers and quantify their impact on belief contagion and risk-return profiles under varying poisoning rates. We then compare alternative GMATS architectures including single-agent, multi-analyst, and two defense-oriented coordinator designs (wide bull-bear and a pairwise-triangulation

coordinator) and show that coordinator-level structure can substantially dampen adversarial shocks and improve robustness under identical poisoning budgets.

Related works. Recent LLM-driven trading frameworks coordinate specialized agents (e.g., fundamental, sentiment, technical, risk) via debate and role decomposition to emulate trading desks. *TradingAgents* formalizes this pattern with analyst teams feeding a decision agent under risk controls, while *TradingGroup* adds structured self-reflection and an end-to-end data-synthesis loop to improve coordination and backtested performance. *ContestTrade* introduces an internal contest mechanism with separate Data and Research teams whose outputs are continuously scored and filtered by real-market feedback, and *QuantAgent* targets high-frequency trading with price-driven Indicator, Pattern, Trend, and Risk agents operating over short-horizon market signals. Together, these systems highlight the importance of inter-agent communication, ranking, and ingestion pipelines that GMATS abstracts and that our input-only attacker seeks to perturb [29,26,33,30]. However, these systems are typically evaluated only on backtested performance under benign data, leaving their robustness to adversarial evidence especially social media and the resulting *contagion dynamics* across agents largely unexplored.

2 Generic Multi-Agent Trading System (GMATS)

Definition 1 (GMATS). (*Architecture in Figure 1.*) Let $\mathcal{A}$ be the asset set ($|\mathcal{A}| = n$) and let $\mathbb{T}$ be a discrete time index set. A GMATS instance is the tuple

$$G = (\mathcal{A}, A, C, K, X, \mathcal{S}, \Sigma, \mathcal{D}, \mathcal{L}, \Lambda, \mathcal{E}),$$

with the following components:

- **Agents.** Analysts A ($|A| \geq 1$), coordinators C ($|C| \geq 1$), controllers K ($|K| \geq 0$), executors X ($|X| \geq 0$). Agents may be LLM-based or programmatic.
- **Typed messages and schema.** $\mathcal{S}$ is a finite set of message types. A message is

$$m = (\text{id}, \text{src}, \text{dst}, \tau, \text{payload}, \text{ts}),$$

with $\tau \in \mathcal{S}$ and timestamp $\text{ts} \in \mathbb{T}$. Messages live on a shared bus; at round $t \in \mathbb{T}$, only messages with $\text{ts} \leq t$ may be consumed (point-in-time, PIT).

- **Schedule.** Σ specifies, for each round t , the order in which agents are activated together with any random seeds used in their calls (e.g., LLM sampling seeds). Each round t executes the pipeline $A \rightarrow C \rightarrow K \rightarrow X$, with micro-rounds permitted inside each stage.
- **Data and environment.** $\mathcal{D} = \{\mathcal{D}_t\}_{t \in \mathbb{T}}$ denotes point-in-time data sources (market, news, social, fundamentals, etc). $\mathcal{E}$ is the execution environment with state space $\mathcal{X}$, which maps orders and current state to fills and next state:

$$(\text{fills}, \chi_{t+1}) = \mathcal{E}(o_t, \chi_t), \quad \chi_t \in \mathcal{X}.$$

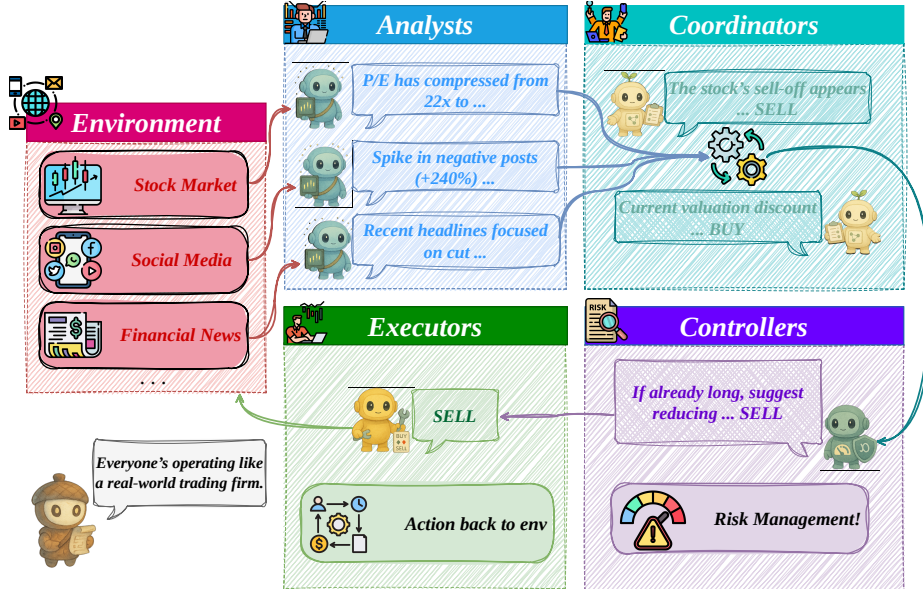


Fig. 1. Architecture of the Generic Multi-Agent Trading System (GMATS).

- **Coordination and control.** $\mathcal{L} = \{\mathcal{L}_t\}_{t \in \mathbb{T}}$ denotes coordinator logic that maps analyst insights and context to stances and draft orders, and $\Lambda = \{\Lambda_t\}_{t \in \mathbb{T}}$ denotes controller logic that enforces risk and constraint policies on these stances or orders. We write ξ_t for any additional context (e.g., portfolio summaries, constraints) supplied to coordinators and controllers at round t .

Contracts (per round t).

1. **Analysts (A).** Each $a \in A$ emits zero or more *insight* messages $m_j^{(a)}$ with type $\tau \in \mathcal{S}$ and $ts \leq t$, computed from PIT data $\mathcal{D}_t$ and the inbox of a .
2. **Coordinators (C).** Consuming all analyst insights at round t , the coordinator logic $\mathcal{L}_t$ produces a *stance*

$$(s_t, \rho_t) = \mathcal{L}_t(\text{insights}_t, \xi_t), \quad s_t \in \mathbb{R}^n,$$

with metadata ρ_t . Optionally, a draft order $o_t \in \mathcal{O}$ may be produced in place of s_t .

3. **Controllers (K).** The risk/constraints gate

$$(\text{ok}_t, o'_t) = \Lambda_t(z_t, \xi_t), \quad z_t \in \{s_t, o_t\}, \quad o'_t \in \mathcal{O},$$

enforces limits (exposure, leverage, liquidity, CVaR/SL/TP, compliance) and yields an executable order o'_t (or $\text{ok}_t = \text{false}$ if no trade is allowed).

4. **Executors (X).** If $|X| > 0$, an executor submits o'_t to the environment:

$$(\text{fills}_t, \chi_{t+1}) = \mathcal{E}(o'_t, \chi_t).$$

If $|X| = 0$, $\mathcal{E}$ is called directly by the coordinator/controller as the final step.

Orders space. Discrete orders live in

$$\mathcal{O}_{\text{disc}} = \{-1, 0, 1\}^n,$$

and continuous portfolio weights live in

$$\mathcal{O}_{\text{cont}} \subseteq \mathbb{R}^n,$$

e.g., with ℓ_1/ℓ_∞ and leverage caps. We write $\mathcal{O} = \mathcal{O}_{\text{disc}} \cup \mathcal{O}_{\text{cont}}$ for the overall order space.

Invariants. (i) **PIT**: all consumed evidence/messages satisfy $t_s \leq t$; (ii) **Replayability**: each run emits a complete log of messages, prompts, LLM responses, and orders, so that the resulting trajectory can be replayed offline for the same Σ without re-querying the LLMs; (iii) **Schema safety**: only well-typed messages ($\tau \in \mathcal{S}$) are routable.

Theorem 1 (Expressivity of GMATS). *Let $\mathcal{M}$ be a multi-agent trading pipeline that satisfies:*

1. **Typed message passing.** *Agents in $\mathcal{M}$ communicate only via messages from a finite schema $\mathcal{S}_{\mathcal{M}}$, and each agent’s behavior is a function of its inbox and point-in-time environment data.*
2. **Point-in-time discipline.** *At round t , every agent in $\mathcal{M}$ may only consume messages and environment data with timestamps $\leq t$.*
3. **Stage separation.** *For each round t , the computation of trading decisions in $\mathcal{M}$ can be decomposed into four phases: (i) evidence ingestion, (ii) fusion and scoring, (iii) constraints and risk checks, and (iv) order execution.*

Then there exists a GMATS instance

$$G = (\mathcal{A}, A, C, K, X, \mathcal{S}, \Sigma, \mathcal{D}, \mathcal{L}, A, \mathcal{E})$$

such that, for any initial environment state and fixed random seeds, the sequence of executed orders produced by $\mathcal{M}$ is identical to the sequence produced by G . We provide the proof in supplementary materials.

Corollary 1 (Compilation of existing frameworks). *TradingAgents, TradingGroup, ContestTrade, and QuantAgent all satisfy the conditions of Theorem 1 and can be compiled into GMATS by mapping their ingestion, fusion, risk, and execution components into the $A \rightarrow C \rightarrow K \rightarrow X$ pipeline and unrolling their internal coordination into micro-rounds of Σ .*

3 Metrics

Setup. We evaluate an *attack* run against a *clean* (no-poison) baseline over T decision times and N tradable assets. Let $\mathcal{P}$ be the set of poisoned messages, and

let $\text{Top}_k(t)$ denote the k highest-ranked items actually consumed by the social analyst at time t . We define the poison-consumption rate

$$p_{\text{poison}} = \frac{1}{T} \sum_{t=1}^T \frac{|\text{Top}_k(t) \cap \mathcal{P}|}{k}, \quad (1)$$

and in Experiments A–C we vary the attacker’s budget until p_{poison} hits target levels (e.g., 15%, 30%, ..., 75%). Unless stated otherwise, performance metrics are reported as *attack minus clean*.

3.1 Contagion Metrics

Belief Signal Shift (BSS). BSS measures how the attack systematically shifts the agent’s internal numeric beliefs, even before trades are placed. Let $f_{i,t}^{\text{attack}}$ and $f_{i,t}^{\text{clean}}$ denote the forecast for asset i at time t under the attacked and clean runs, respectively. We define the aggregate belief shift as

$$\text{BSS} = \frac{1}{TN} \sum_{t=1}^T \sum_{i=1}^N (f_{i,t}^{\text{attack}} - f_{i,t}^{\text{clean}}). \quad (2)$$

A positive BSS indicates that, on average, forecasts become more bullish under attack, while a negative BSS indicates a net shift toward more bearish beliefs.

3.2 Performance Metrics (FinSABER; attack–clean)

Protocol. We adopt the FinSABER performance protocol as described in [14]. Let $\{r_t\}_{t=1}^T$ be the net (after-cost) portfolio returns at the chosen frequency (daily by default). Define A as the annualization factor ($A=252$ for daily).

Cumulative and annualized return. These metrics quantify how much the portfolio grows in total (CR) and what that growth corresponds to as an annualized rate (AR), under attack versus clean.

$$\text{CR} = \prod_{t=1}^T (1 + r_t) - 1, \quad \text{AR} = (1 + \text{CR})^{A/T} - 1. \quad (3)$$

Annualized volatility. Annualized volatility captures the scale of return fluctuations, i.e., how “noisy” or risky the strategy is over a year.

$$\text{AV} = \sqrt{A} \text{sd}(r_t). \quad (4)$$

Sharpe ratio. Sharpe summarizes risk-adjusted performance using total volatility [22].

$$\text{SR} = \frac{\sqrt{A} \text{mean}(r_t)}{\text{sd}(r_t)}. \quad (5)$$

Delta reporting. To isolate the effect of the attack, we report performance as the difference between attack and clean runs. For each metric $m \in \{\text{CR}, \text{AV}, \text{SR}\}$, we compute

$$\Delta m = m^{\text{attack}} - m^{\text{clean}}. \quad (6)$$

4 Attacker Model

Attacker taxonomy We consider two simple but representative attack policies. Both respect the same daily budget B on the single target asset. **random attacker** serves as a minimal baseline. On each day t it spends its budget B by generating posts with random sentiment directions, independent across posts and days (e.g., sampling bullish vs. bearish with equal probability). This attacker injects noise without targeting and any contagion it induces reflects a general sensitivity of GMATS to low-level social perturbations. The **always-negative attacker** is a simple one-sided stressor. On each day t it spends its entire budget B generating strongly bearish posts about the target asset. This policy approximates a persistent negative-narrative campaign (e.g., a “short-and-distort” style effort) that continuously pushes sentiment in the same direction. Together, these two attackers span a simple design space: untargeted, sign-symmetric noise (random) and untargeted but persistently one-sided pressure (always-negative). Our experiments in Section 5 compare their induced contagion profiles under common budgets B and varying p_{poison} .

Scope and admission assumption. We measure *downstream contagion conditioned on adversarial content being present in the social evidence stream*, rather than modeling how such content is created, promoted, or filtered. Because real trading stacks differ widely in post scoring/admission (heuristics, sentiment models, LLM filters), an end-to-end seeding model would confound results with system-specific details. We therefore assume attacker posts are *admissible* and vary attack strength via the *effective poison rate* p_{poison} , defined as the fraction of *consumed* (top- k) posts attributable to the attacker, which isolates susceptibility under a fixed retriever. This is realistic on open social channels where volume can surface some posts without retriever compromise [6,27], and where pipelines often upweight high-salience or strongly valenced content [18,24,34]. Thus p_{poison} serves as an abstract knob capturing posting intensity and any upstream amplification while keeping evaluation focused on what happens *after* poisoned posts enter the system.

5 Experiments

5.1 Data and Setup

We instantiate GMATS with three analyst roles (social, fundamentals, news), one coordinator, one risk controller, and one executor. All experiments use `gpt-4o-mini` and ran over the same historical window (**01 Nov 2021–01 Dec**

Table 1. Experiment A (GMATS, fixed architecture): results are attack-clean deltas across fraction of consumed posts that are poisoned, p_{poison} . Values are averaged across 5 runs.

Metric	15%	30%	45%	60%	75%
Random Attacker					
BSS _{analyst}	+0.077	+0.163	+0.214	+0.252	+0.280
BSS _{coord}	+0.091	+0.269	+0.270	+0.332	+0.360
ΔCR	+0.020	+0.009	+0.023	+0.016	+0.017
ΔSharpe	+0.360	+1.080	+2.040	+0.998	+0.600
ΔVol	+0.082	+0.036	+0.051	+0.104	+0.150
Always Negative Attacker					
BSS _{analyst}	-0.777	-1.084	-1.204	-1.291	-1.413
BSS _{coord}	-0.409	-0.544	-0.544	-0.642	-0.594
ΔCR	-0.006	-0.032	-0.036	-0.043	-0.044
ΔSharpe	+0.864	-5.501	-5.970	-6.167	-6.060
ΔVol	+0.005	+0.010	+0.020	+0.035	+0.046

2021) for AAPL. We focus on a single ticker/month for controlled mechanistic analysis. Extending to more tickers or months primarily increases sample size (tightening uncertainty) but does not change the propagation mechanism captured by our contagion metrics under the same PIT evidence-admission and poisoning budget. We also release a one-year corpus of equity-related social media posts collated from tweets and reddit for the top 7 large-cap tech companies (AAPL, AMZN, GOOG, META, MSFT, TSLA, NFLX) spanning 30 Sept 2021 – 29 Sept 2022. Each record provides a timestamp, tweet text, ticker, and company name. We build an offline environment using historical equity prices. GMATS agents operate strictly in PIT mode, consuming only the data available at each round preventing leakage of future information [7].

5.2 Experiment A: Contagion vs Poisoning Rate in GMATS

Experiment A asks: *how much contagion and damage can an input-only attacker cause in a fixed GMATS instance as we increase the share of poisoned evidence seen by the system?*

Table 1 summarizes the dose-response across attacker type and realized poisoning rate ($p_{\text{poison}} \in \{15, 30, 45, 60, 75\}\%$). For each (attacker, p_{poison}) pair we report attack-clean deltas for belief-contagion metrics and trading performance.

Across metrics, we observe a clear dose-response in belief shift. For both attackers, the absolute value of BSS increases with p_{poison} , but the effect is much stronger for the Always Negative attacker: BSS_{analyst} moves from roughly -0.78 at 15% to about -1.41 at 75%, while BSS_{coord} tracks a smaller but still substantial negative shift. The coordinator is therefore less sensitive than the social analyst, reflecting partial damping by the fundamentals and news analysts, but the negative bias propagates through the stack rather than being fully absorbed.

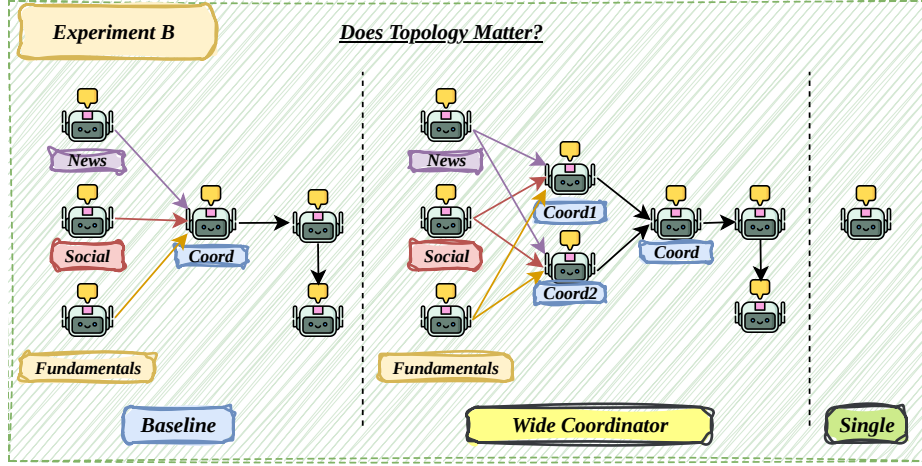


Fig. 2. Experiment B (architectural contagion). Three GMATS realizations with identical tools but different topologies.

On the performance side, the Random attacker produces only mild changes: ΔCR and $\Delta Sharpe$ remain small and sometimes slightly positive, suggesting that low-rate, directionally symmetric noise in the social channel does not reliably harm the system and may occasionally coincide with beneficial risk adjustments. In contrast, the Always Negative attacker systematically degrades sharpe and cumulative return as p_{poison} increases. There is a pronounced regime change around 30%–45% ingestion: $\Delta Sharpe$ flips from mildly positive at 15% to sharply negative (around -6), while volatility also worsen. Overall, Experiment A indicates that GMATS is relatively robust to low-level perturbations but becomes highly vulnerable once a single channel is populated with sufficiently dense, directional adversarial signals. Sharpe is annualized and can be sensitive to short evaluation windows [17,2], as a result, absolute Sharpe levels in our one-month study are not intended as standalone performance claims. Our conclusions are based on *attack-clean deltas* (including the direction/sign of $\Delta Sharpe$) and are cross-checked against ΔCR and ΔVol . We also ran Experiment A on TradingAgents, the full results are reported in the supplementary materials.

5.3 Experiment B: Architectural Contagion Across Topologies

Experiment B asks a complementary question: *does a multi-agent message-passing topology amplify adversarial shocks compared to a single-agent system?* To isolate architectural effects, we study two additional realizations of GMATS with identical tools, prompts, and budgets but different message-passing graphs (Figure 2):

Table 2. Experiment B (GMATS, Topology): attack-clean deltas across fraction of consumed posts that are poisoned, p_{poison} . Values are averaged across 5 runs.

Metric	15%	30%	45%	60%	75%
Wide Coordinator					
BSS _{analyst}	-0.754	-0.964	-1.136	-1.213	-1.291
BSS _{coord1}	-0.271	-0.339	-0.455	-0.478	-0.467
BSS _{coord2}	-0.284	-0.356	-0.470	-0.506	-0.471
BSS _{fuse}	-0.281	-0.353	-0.471	-0.499	-0.480
ΔCR	-0.003	-0.014	-0.022	-0.024	-0.014
ΔSharpe	+3.235	+0.436	-1.980	-3.113	-0.062
ΔVol	-0.005	-0.012	-0.010	-0.015	-0.007
Single Agent					
BSS _{analyst}	-1.148	-1.364	-1.415	-1.491	-1.386
ΔCR	-0.052	-0.056	-0.057	-0.062	-0.054
ΔSharpe	-7.153	-7.150	-7.274	-6.773	-8.670
ΔVol	+0.007	+0.015	+0.019	+0.043	+0.015

- **Wide coordinator (multi-agent):** the three analysts feed into two coordinators with cross-talk before a shared controller, increasing the number of aggregation hops and inter-agent interactions.
- **Single agent:** a monolithic agent that directly ingests social, news, and fundamental streams and outputs portfolio weights, with no internal analyst-coordinator decomposition.

Table 2 reveals that architecture has a first-order effect on both belief shift and trading damage. Under the Single Agent baseline, the Always Negative Attacker induces consistently larger-magnitude belief shifts and substantially worse performance: BSS_{analyst} is around -1.1 to -1.5 across poisoning rates, and the corresponding ΔSharpe remains strongly negative (roughly -7 to -9) with sizeable drops in cumulative return compared to the wide coordinator.

In contrast, the Wide Coordinator topology attenuates the attack. The social analyst still becomes increasingly bearish as p_{poison} grows, but the downstream coordinator ensemble (coord1, coord2, and their fused view) exhibits smaller-magnitude BSS values, indicating partial damping of the contaminated social signal by the additional analysts and coordination stages. Trading impact is also milder: ΔCR remains close to zero, ΔSharpe is even positive at low poisoning rates and only moderately negative at higher rates, and volatility delta stay much closer to the clean baseline. Overall, Experiment B suggests that multi-agent message passing with heterogeneous information sources can *dampen*, rather than amplify, adversarial shocks.

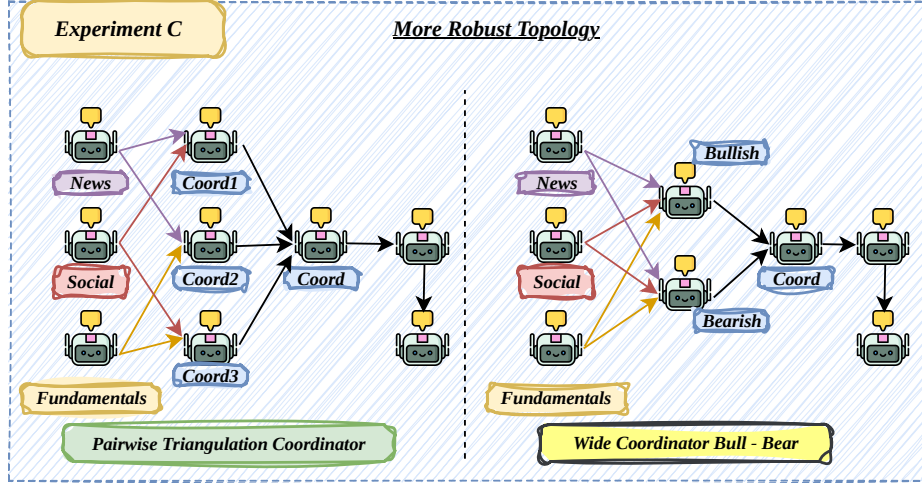


Fig. 3. Experiment C (robust coordinators). Two defense-oriented coordinator designs—pairwise triangulation and wide bull–bear—aimed at making GMATS more robust to adversarial social signals.

5.4 Experiment C: Robust Architectures for Contagion Mitigation

Experiment C turns to defenses: *given a fixed attacker, how far can we push down contagion and trading damage by redesigning the coordinator layer?*

We modify only the coordinator topology, instantiating two defense-oriented designs (Figure 3):

- **Wide Coordinator Bull–Bear.** Two coordinators receive the same analyst inboxes but are instructed via prompts and routing to focus exclusively on either bullish or bearish signals. A fuse coordinator then aggregates their reports and scorecards into a final stance. This forces the system to explicitly surface and reconcile opposing narratives [5] instead of allowing a single coordinator to ignore minority evidence.
- **Pairwise Triangulation Coordinator.** Three coordinators with partially overlapping mandates read the same analyst outputs. Their views are then fused by a meta-coordinator that is encouraged to downweight outliers and reward pairwise agreement, yielding a more conservative aggregate stance when the inputs disagree.

As in Table 3, in the **Wide Coordinator Bull–Bear** design, the bull and bear specialists each move only modestly in the negative direction, and the fused coordinator stays in an intermediate regime (e.g., BSS_{fuse} between about -0.4 and -0.6 across poisoning rates). This suggests that forcing separate bullish and bearish views limits the extent to which a unidirectional attack can fully dominate the final stance. Trading impact is likewise muted: ΔCR remains very close to zero, volatility and drawdown deltas stay small, and ΔSharpe is *positive* at all

Table 3. Experiment C (GMATS, Defense): attack-clean deltas across fraction of consumed posts that are poisoned, p_{poison} . Values are averaged across 5 runs.

Metric	15%	30%	45%	60%	75%
Wide Coordinator Bull-Bear					
$\text{BSS}_{\text{analyst}}$	-0.668	-0.895	-0.980	-0.983	-1.057
BSS_{bull}	-0.138	-0.184	-0.208	-0.219	-0.211
BSS_{bear}	-0.036	-0.074	-0.085	-0.081	-0.084
BSS_{fuse}	-0.403	-0.568	-0.559	-0.552	-0.590
ΔCR	-0.005	-0.006	-0.001	-0.006	-0.013
ΔSharpe	+1.871	+2.524	+5.513	+3.234	+0.252
ΔVol	+0.002	-0.003	-0.006	+0.000	-0.006
Pairwise Triangulation Coordinator					
$\text{BSS}_{\text{analyst}}$	-0.790	-1.024	-1.100	-1.185	-1.275
BSS_1	-0.368	-0.502	-0.538	-0.571	-0.615
BSS_2	-0.523	-0.622	-0.678	-0.724	-0.759
BSS_3	-0.029	-0.024	-0.043	-0.038	-0.026
BSS_{fuse}	-0.314	-0.389	-0.405	-0.403	-0.397
ΔCR	+0.001	-0.006	-0.011	-0.010	-0.008
ΔSharpe	+2.347	+1.586	+1.069	+0.359	+1.713
ΔVol	+0.010	+0.001	-0.005	+0.011	+0.005

poisoning levels, often substantially so. Under attack, this coordinator behaves like a regularizer, improving risk-adjusted performance instead of collapsing as in the naive coordinator from Experiment A.

The **Pairwise Triangulation Coordinator** goes further in damping contagion at the fused output. Although two of the internal coordinators (1 and 2) exhibit sizeable negative BSS values, the third stays near zero and the fused BSS is consistently less negative than any individual coordinator and than the baseline coordinator in Experiment A. This indicates that the triangulation rule is successfully downweighting extreme, attack-aligned views. Again, trading performance under attack is stable: cumulative-return deltas are close to zero, draw-down changes are small, and ΔSharpe remains positive across all poisoning rates.

Taken together, Experiment C shows that coordinator-level defenses can substantially reduce effective contagion at the decision layer and prevent the Sharpe collapse observed in earlier experiments. Rather than being inherently fragile, multi-LLM architectures can be configured so that additional coordination structure acts as a form of robustness.

Qualitative contagion traces. To complement the aggregate deltas in Tables 2–3, we inspect a representative *contagion trace* for AAPL on **2021-11-05** (Figure 4). Each panel shows the internal message chain from intermediate coordinators to the final fused stance, together with the resulting $(\mu, \sigma, \text{confidence})$ scorecards. Across coordinator designs, the same directional pressure from the social channel leads to qualitatively different decision-layer outcomes: (i) **Wide coordinator**

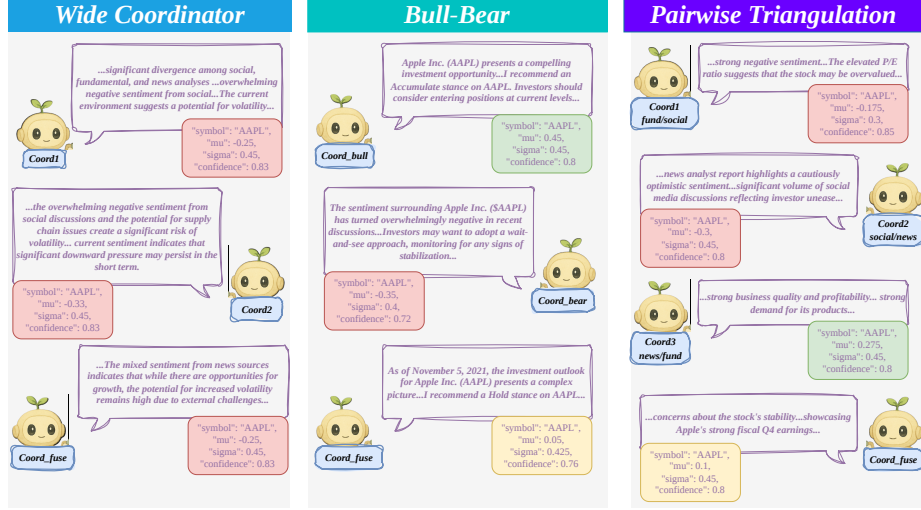


Fig. 4. Experimental Traces. Bubbles are truncated for space.

produces consistently bearish intermediate stances and a bearish fuse ($\mu < 0$), indicating that the negative social narrative largely survives aggregation. (ii) **Wide bull–bear** forces an explicit pro/con decomposition: the bull coordinator remains strongly positive while the bear coordinator turns negative, and the fuse lands near neutral ($\mu \approx 0$), yielding a “hold”-like stance rather than a collapse into the attacker’s direction. (iii) **Pairwise triangulation** mixes channels before fusion (fund/social, social/news, news/fund), surfacing cross-channel disagreements. The final fuse stays mildly positive ($\mu > 0$) despite negative social evidence, consistent with the quantitative finding that defense-oriented coordinators reduce effective contagion at the decision layer.

5.5 Model Sensitivity: Replication with Qwen2.5-7B-Instruct

To test whether our conclusions depend on a single LLM backend, we replicate the core GMATS experiments with **Qwen2.5-7B-Instruct** [31]. We keep the ticker, historical window, PIT data protocol, attacker policies, poisoning-rate targets, prompts, and topologies fixed, changing only the model backend. Table 4 reports attack–clean deltas averaged over 5 runs.

The Qwen replication preserves the main qualitative findings. First, directional poisoning remains substantially more damaging than random sentiment injection. Under the Random attacker, performance deltas remain small and often positive. Under the Always Negative attacker, the baseline GMATS coordinator shifts sharply negative: BSS_{analyst} ranges from -1.159 to -1.694 , BSS_{coord} ranges from -0.554 to -0.871 , ΔCR decreases from -0.015 to -0.061 , and ΔSharpe falls to roughly -5 to -6 once $p_{\text{poison}} \geq 30\%$. This matches the main

Table 4. Additional experiment results under **Qwen2.5-7B-Instruct** on GMATS variants. Entries are attack-clean deltas averaged over 5 runs.

Random Attacker					
Metric	15%	30%	45%	60%	75%
BSS _{analyst}	-0.368	-0.353	-0.365	-0.515	-0.341
BSS _{coord}	-0.403	-0.457	-0.511	-0.395	-0.439
Δ CR	+0.010	+0.024	+0.005	+0.016	+0.005
Δ Sharpe	+0.092	+0.313	+1.969	+1.092	+1.448
Δ Vol	-0.019	+0.011	+0.012	+0.024	+0.024

Wide Coordinator Bull-Bear					
Metric	15%	30%	45%	60%	75%
BSS _{analyst}	-0.873	-1.123	-1.149	-1.347	-1.313
BSS _{bull}	-0.562	-0.596	-0.630	-0.689	-0.637
BSS _{bear}	+0.008	-0.089	-0.072	-0.171	-0.150
BSS _{fuse}	-0.293	-0.335	-0.333	-0.409	-0.369
Δ CR	+0.004	-0.001	-0.005	-0.005	-0.009
Δ Sharpe	+3.373	+2.660	+1.341	+0.354	-0.115
Δ Vol	+0.000	-0.002	-0.004	+0.010	+0.000

Single Agent					
Metric	15%	30%	45%	60%	75%
BSS _{analyst}	-1.459	-1.805	-1.876	-1.902	-1.859
Δ CR	-0.045	-0.046	-0.047	-0.052	-0.047
Δ Sharpe	-6.369	-6.342	-6.242	-6.714	-7.546
Δ Vol	+0.002	+0.013	+0.018	+0.028	+0.019
—	—	—	—	—	—
—	—	—	—	—	—
—	—	—	—	—	—
—	—	—	—	—	—

Always Negative Attacker					
Metric	15%	30%	45%	60%	75%
BSS _{analyst}	-1.159	-1.606	-1.594	-1.557	-1.694
BSS _{coord}	-0.554	-0.815	-0.782	-0.835	-0.871
Δ CR	-0.015	-0.036	-0.047	-0.054	-0.061
Δ Sharpe	+0.227	-5.062	-5.350	-5.873	-6.002
Δ Vol	+0.008	+0.012	+0.014	+0.023	+0.046

Wide Coordinator					
Metric	15%	30%	45%	60%	75%
BSS _{analyst}	-0.798	-0.963	-1.119	-1.135	-1.217
BSS _{coord1}	-0.579	-0.697	-0.858	-0.868	-0.777
BSS _{coord2}	-0.632	-0.760	-0.911	-0.985	-0.938
BSS _{fuse}	-0.634	-0.764	-0.916	-0.939	-0.879
Δ CR	-0.015	-0.016	-0.021	-0.019	-0.019
Δ Sharpe	+0.240	-0.210	-2.012	-0.413	-1.100
Δ Vol	-0.011	-0.012	-0.012	-0.018	-0.010

Pairwise Triangulation					
Metric	15%	30%	45%	60%	75%
BSS _{analyst}	-0.982	-1.241	-1.296	-1.468	-1.522
BSS ₁	-0.412	-0.557	-0.601	-0.655	-0.698
BSS ₂	-0.584	-0.702	-0.757	-0.803	-0.839
BSS ₃	-0.031	-0.028	-0.041	-0.037	-0.030
BSS _{fuse}	-0.247	-0.318	-0.336	-0.341	-0.334
Δ CR	+0.002	-0.003	-0.007	-0.008	-0.007
Δ Sharpe	+2.814	+2.031	+1.124	+0.462	+0.736
Δ Vol	+0.008	+0.002	-0.003	+0.009	+0.004

gpt-4o-mini conclusion that dense, one-sided social evidence can induce both belief contagion and portfolio-level degradation.

Second, the architectural ranking is stable. The Single Agent variant is consistently brittle, with Δ Sharpe between approximately -6.2 and -7.5 . The Wide Coordinator still suffers belief shifts, but its trading damage is smaller, with Δ CR close to zero and substantially milder Sharpe degradation. Finally, both defense-oriented topologies remain robust at the fused decision layer. Wide Coordinator Bull-Bear keeps BSS_{fuse} between -0.293 and -0.409 , while Pairwise Triangulation keeps BSS_{fuse} between -0.247 and -0.341 and maintains positive Δ Sharpe across all poisoning rates. Thus, across two LLM backends, the same qualitative pattern holds: one-sided social poisoning is the damaging case, single-agent aggregation is most brittle, and coordinator designs that explicitly separate or triangulate conflicting evidence reduce downstream damage.

At the same time, the exact magnitudes are model-dependent. Qwen exhibits stronger negative shifts in some intermediate coordinator outputs than **gpt-4o-mini**, especially in the Wide Coordinator topology. We therefore interpret BSS and Sharpe deltas as model-conditional stress-test measurements, while treating the cross-model consistency of the architectural ordering as the stronger conclusion.

6 Discussion

We deliberately restricted our study to scripted, input-only attackers that are non-adaptive, single-channel (social), and evaluated under offline replay. An immediate extension is to stress-test GMATS over longer horizons and diverse market regimes (including prolonged bear markets), where a one-sided narrative can sometimes align with fundamentals and yield qualitatively different effects. Building on [4], an RL attacker could be embedded in GMATS to steer an LLM post generator (e.g., sentiment/timing/style under a budget) and optimize a contagion-aware objective rather than purely price-based rewards. One simple form is

$$r_t = -\alpha \frac{\text{PnL}_t^{\text{attack}} - \text{PnL}_t^{\text{clean}}}{\text{scale}_t} + \beta_A \text{BSS}_{\text{analyst}} + \beta_C \text{BSS}_{\text{coord}}, \quad (7)$$

where $\text{scale}_t = \max(\epsilon, \text{EMA}(|\text{PnL}_t^{\text{attack}} - \text{PnL}_t^{\text{clean}}|))$ stabilizes learning. Evaluating such adaptive policies and how coordinator-level defenses shift the resulting utility–security frontier is an important next step toward characterizing worst-case brittleness in multi-agent trading stacks.

Beyond sentiment manipulation, an orthogonal threat is *instruction hijacking*: untrusted posts may embed prompt injections that subvert role constraints or induce unintended tool use [16,32,12]. In multi-agent pipelines, a single compromised analyst output can cascade through fusion and affect trading decisions. Future work could benchmark injection success rates in GMATS under realistic retrieval, and test lightweight mitigations (instruction/data separation, tool gating, and cross-agent consistency checks).

7 Conclusion

We studied how adversarial social signals propagate through LLM-native multi-agent trading stacks. We introduced the Generic Multi-Agent Trading System (GMATS) and proposed contagion metrics that trace how poisoned social inputs shift beliefs at analyst and coordinator layers and impact downstream risk–return. Experiments show that these systems are not inherently robust. Injecting systematically negative posts can cause large belief shifts and materially worsen returns. We also find certain types of topologies and designs can dampen contagion and improve robustness under attack. Our study is limited to a narrow set of assets and horizons but is designed as a controlled, reproducible starting point. We hope that GMATS, together with our contagion metrics and experimental protocol, can serve as a foundation for future work on attacks and defenses in multi-agent trading systems, including stronger multi-channel attackers and principled defenses at both architectural and prompt levels.

Acknowledgments. This research/project is supported by the National Research Foundation, Singapore under its AI Singapore Programme (AISG Award No: AISG3-PhD/2025-08-063).

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

Data Collection and Usage Reddit posts were collected from publicly available Pushshift Reddit dumps. Twitter data were sourced from a public Kaggle dataset; we do not redistribute tweet text and release only data as permitted.

References

1. Antweiler, W., Frank, M.Z.: Is all that talk just noise? the information content of internet stock message boards. *Journal of Finance* **59**(3), 1259–1294 (2004). <https://doi.org/10.1111/j.1540-6261.2004.00662.x>
2. Bailey, D.H., López de Prado, M.: The deflated sharpe ratio: Correcting for selection bias, backtest overfitting and non-normality. *The Journal of Portfolio Management* **40**(5), 94–107 (2014). <https://doi.org/10.3905/jpm.2014.40.5.094>
3. Bollen, J., Mao, H., Zeng, X.: Twitter mood predicts the stock market. *Journal of Computational Science* **2**(1), 1–8 (Mar 2011). <https://doi.org/10.1016/j.jocs.2010.12.007>, <https://www.sciencedirect.com/science/article/abs/pii/S187775031100007X>
4. Byrd, D.: Exploring sentiment manipulation by llm-enabled intelligent trading agents (2025). <https://doi.org/10.48550/arXiv.2502.16343>, <https://arxiv.org/abs/2502.16343>, arXiv:2502.16343 (v1), 2025-02-22
5. Estornell, A., Liu, Y.: Multi-llm debate: Framework, principals, and interventions. In: *Advances in Neural Information Processing Systems*. vol. 37 (2024). <https://doi.org/10.52202/079017-0911>, https://proceedings.neurips.cc/paper_files/paper/2024/hash/32e07a110c6c6acff1afbf2bf82b614ad-Abstract-Conference.html
6. Ferrara, E., Varol, O., Davis, C., Menczer, F., Flammini, A.: The rise of social bots. *Communications of the ACM* **59**(7), 96–104 (Jul 2016). <https://doi.org/10.1145/2818717>, <https://cacm.acm.org/research/the-rise-of-social-bots/>
7. Glasserman, P., Lin, C.: Assessing look-ahead bias in stock return predictions generated by gpt sentiment analysis (2023). <https://doi.org/10.48550/arXiv.2309.17322>
8. Gu, X., Zheng, X., Pang, T., Du, C., Liu, Q., Wang, Y., Jiang, J., Lin, M.: Agent smith: A single image can jailbreak one million multimodal llm agents exponentially fast (2024). <https://doi.org/10.48550/arXiv.2402.08567>, <https://arxiv.org/abs/2402.08567>, arXiv:2402.08567 (v2), last revised 2024-06-03; ICML 2024
9. Guo, T., Chen, X., Wang, Y., Chang, R., Pei, S., Chawla, N.V., Wiest, O., Zhang, X.: Large language model based multi-agents: A survey of progress and challenges (2024). <https://doi.org/10.48550/arXiv.2402.01680>
10. Hong, S., Zhuge, M., Chen, J., Zheng, X., Cheng, Y., Wang, J., Zhang, C., Wang, Z., Yau, S.K.S., Lin, Z., Zhou, L., Ran, C., Xiao, L., Wu, C., Schmidhuber, J.: MetaGPT: Meta programming for a multi-agent collaborative framework. In: *The Twelfth International Conference on Learning Representations* (2024), <https://openreview.net/forum?id=VtmBAGCN7o>
11. Kogan, S., Moskowitz, T.J., Niessner, M.: Social media and financial news manipulation. *Review of Finance* **27**(4), 1229–1268 (2023). <https://doi.org/10.1093/rof/rfac058>, published online 2022-10-21; Issue July 2023

12. Lee, D., Tiwari, M.: Prompt infection: Llm-to-llm prompt injection within multi-agent systems (2024). <https://doi.org/10.48550/arXiv.2410.07283>
13. Li, G., Hammoud, H., Itani, H., Khizbullin, D., Ghanem, B.: Camel: Communicative agents for “mind” exploration of large language model society. In: Advances in Neural Information Processing Systems. vol. 36, pp. 51991–52008. Curran Associates, Inc. (2023). <https://doi.org/10.48550/arXiv.2303.17760>, https://proceedings.neurips.cc/paper_files/paper/2023/hash/a3621ee907def47c1b952ade25c67698-Abstract-Conference.html
14. Li, W., Kim, H., Cucuringu, M., Ma, T.: Can llm-based financial investing strategies outperform the market in long run? (2025). <https://doi.org/10.48550/arXiv.2505.07078>, <https://arxiv.org/abs/2505.07078>
15. Liu, C., Low, A., Putnins, T.: The real impacts of public short campaigns: Evidence from stakeholders. *Journal of Corporate Finance* **88**, 102624 (2024). <https://doi.org/10.1016/j.jcorpfin.2024.102624>, <https://www.sciencedirect.com/science/article/pii/S0929119924000865>, open-access version via UTS OPUS (published version PDF).
16. Liu, Y., Deng, G., Li, Y., Wang, K., Wang, Z., Wang, X., Zhang, T., Liu, Y., Wang, H., Zheng, Y., Zhang, L.Y., Liu, Y.: Prompt injection attack against llm-integrated applications (2023). <https://doi.org/10.48550/arXiv.2306.05499>, houYi attack; revised 2025-12-29 on arXiv
17. Lo, A.W.: The statistics of sharpe ratios. *Financial Analysts Journal* **58**(4), 36–52 (2002). <https://doi.org/10.2469/faj.v58.n4.2453>
18. Milli, S., Carroll, M., Wang, Y., Pandey, S., Zhao, S., Dragan, A.D.: Engagement, user satisfaction, and the amplification of divisive content on social media. *PNAS Nexus* **4**(3), pgaf062 (2025). <https://doi.org/10.1093/pnasnexus/pgaf062>
19. Nam, D., Skillicorn, D.B.: Detecting pump & dump stock market manipulation from online forums. *Digital Finance* **7**(1), 1–20 (Mar 2025). <https://doi.org/10.1007/s42521-024-00121-4>, <https://link.springer.com/article/10.1007/s42521-024-00121-4>
20. Pohl, J., Griesbach, M., Samiei, A., de Araujo, A.: Social bots spreading disinformation about finance: Research trends, and ethical challenges. In: Code and Conscience: Exploring Technology, Human Rights, and Ethics in Multidisciplinary AI Education. *Lecture Notes in Computer Science*, Springer (2024). https://doi.org/10.1007/978-3-031-52082-2_6
21. Qian, C., Liu, W., Liu, H., Chen, N., Dang, Y., Li, J., Yang, C., Chen, W., Su, Y., Cong, X., Xu, J., Li, D., Liu, Z., Sun, M.: ChatDev: Communicative agents for software development. In: Ku, L.W., Martins, A., Srikumar, V. (eds.) *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. pp. 15174–15186. Association for Computational Linguistics, Bangkok, Thailand (Aug 2024). <https://doi.org/10.18653/v1/2024.acl-long.810>, <https://aclanthology.org/2024.acl-long.810/>
22. Sharpe, W.F.: Mutual fund performance. *The Journal of Business* **39**(1), 119–138 (1966). <https://doi.org/10.1086/294846>
23. Sprenger, T.O., Tumasjan, A., Sandner, P.G., Welpe, I.M.: Tweets and trades: The information content of stock microblogs. *European Financial Management* **20**(5), 926–957 (2014). <https://doi.org/10.1111/j.1468-036X.2013.12007.x>, <https://doi.org/10.1111/j.1468-036X.2013.12007.x>
24. Su, J., Nakov, P., Cardie, C.: Corpus poisoning via approximate greedy gradient descent. In: *Findings of the Association for Computational Linguistics: ACL 2025*. pp. 4274–4294 (2025), <https://aclanthology.org/2025.findings-acl.222/>

25. Tetlock, P.C.: Giving content to investor sentiment: The role of media in the stock market. *The Journal of Finance* **62**(3), 1139–1168 (2007). <https://doi.org/10.1111/j.1540-6261.2007.01232.x>, https://www.columbia.edu/~pt2238/papers/Tetlock_Media_Sentiment_JF.pdf, journal article; commonly circulated as a PDF preprint/author version.
26. Tian, F., Salim, F.D., Xue, H.: Tradinggroup: A multi-agent trading system with self-reflection and data-synthesis (2025). <https://doi.org/10.48550/arXiv.2508.17565>, <https://arxiv.org/abs/2508.17565>, arXiv:2508.17565 (v1), 2025-08-25
27. Vykopal, I., Pikuliak, M., Srba, I., Moro, R., Macko, D., Bielikova, M.: Disinformation capabilities of large language models. In: *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. pp. 14830–14847 (2024). <https://doi.org/10.18653/v1/2024.acl-long.793>, <https://aclanthology.org/2024.acl-long.793/>
28. Wu, Q., Bansal, G., Zhang, J., Wu, Y., Li, B., Zhu, E., Jiang, L., Zhang, X., Zhang, S., Liu, J., Awadallah, A.H., White, R.W., Burger, D., Wang, C.: Autogen: Enabling next-gen LLM applications via multi-agent conversation (2023). <https://doi.org/10.48550/arXiv.2308.08155>, <https://arxiv.org/abs/2308.08155>, arXiv:2308.08155 (v2), last revised 2023-10-03
29. Xiao, Y., Sun, E., Luo, D., Wang, W.: Tradingagents: Multi-agents llm financial trading framework (2025). <https://doi.org/10.48550/arXiv.2412.20138>, <https://arxiv.org/abs/2412.20138>, arXiv:2412.20138 (v7), last revised 2025-06-03
30. Xiong, F., Zhang, X., Feng, A., Sun, S., You, C.: Quantagent: Price-driven multi-agent llms for high-frequency trading (2025). <https://doi.org/10.48550/arXiv.2509.09995>, <https://arxiv.org/abs/2509.09995>, arXiv:2509.09995 (v3), last revised 2025-09-27
31. Yang, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Li, C., Liu, D., Huang, F., Wei, H., Lin, H., Yang, J., Tu, J., Zhang, J., Yang, J., Yang, J., Zhou, J., Lin, J., Dang, K., Lu, K., Bao, K., Yang, K., Yu, L., Li, M., Xue, M., Zhang, P., Zhu, Q., Men, R., Lin, R., Li, T., Tang, T., Xia, T., Ren, X., Ren, X., Fan, Y., Su, Y., Zhang, Y., Wan, Y., Liu, Y., Cui, Z., Zhang, Z., Qiu, Z.: Qwen2.5 technical report (2025), <https://arxiv.org/abs/2412.15115>
32. Zhan, Q., Liang, Z., Ying, Z., Kang, D.: Injecagent: Benchmarking indirect prompt injections in tool-integrated large language model agents (2024). <https://doi.org/10.48550/arXiv.2403.02691>, findings of ACL 2024
33. Zhao, L., Sun, R., Jiang, Z., Yang, B., Bai, Y., Chen, M., Wang, X., Li, J., Bai, Z.: Contesttrade: A multi-agent trading system based on internal contest mechanism (2025). <https://doi.org/10.48550/arXiv.2508.00554>, <https://arxiv.org/abs/2508.00554>, arXiv:2508.00554 (v3), last revised 2025-08-18
34. Zou, W., Geng, R., Wang, B., Jia, J.: Poisonedrag: Knowledge corruption attacks to retrieval-augmented generation of large language models. In: *Proceedings of the 34th USENIX Security Symposium (USENIX Security '25)*. pp. 3827–3844. USENIX Association, Seattle, WA, USA (Aug 2025), <https://www.usenix.org/conference/usenixsecurity25/presentation/zou-poisonedrag>

Contagion on the Trading Floor: How Adversarial Signals Spread in Multi-Agent Trading Systems (Supplementary Material)

Qi Rong Sua¹, Junhao Dong^{1,3}, Nguyen Duc Thai¹, Yuqing Wen², Cheston Tan³, and Yew-Soon Ong^{1,3} (✉)

¹ Nanyang Technological University, Singapore
{suaq0001,junhao003,ducthai001,asysong}@ntu.edu.sg

² National University of Singapore, Singapore
e1351400@u.nus.edu

³ Centre for Frontier AI Research, IHPC, A*STAR, Singapore
cheston_tan@a-star.edu.sg

Abstract. This supplementary material complements the main paper in two ways. First, it gives the full proof of Theorem 1 (Expressivity of GMATS), showing that any multi-agent trading pipeline satisfying typed message passing, point-in-time (PIT) discipline, and stage separation can be simulated by a GMATS instance that produces the same executed-order sequence under matched randomness. Second, it gives a concrete compilation recipe for mapping modern agentic trading frameworks into GMATS by aligning their ingestion, fusion, risk, and execution components with the $A \rightarrow C \rightarrow K \rightarrow X$ pipeline and unrolling internal debate, contest, and reflection steps into micro-rounds of Σ . We also retain an additional *TradingAgents* baseline under the Always Negative attacker. All empirical entries are averaged over 5 runs and reported as attack-clean deltas.

1 Proof of Theorem 1

Proof (Proof of Theorem 1). Let $\mathcal{M}$ be a multi-agent trading pipeline satisfying the three assumptions in Theorem 1. Fix the asset universe, the PIT data stream, an initial environment state χ_1 , and a source of randomness ω . We view every stochastic call in $\mathcal{M}$, including LLM sampling and any stochastic scheduler choice, as consuming a contiguous segment of ω . The goal is to construct a GMATS instance

$$G = (\mathcal{A}, A, C, K, X, \mathcal{S}, \Sigma, \mathcal{D}, \mathcal{L}, A, \mathcal{E})$$

that, under the same χ_1 and ω , executes exactly the same order at every decision time.

Operational View of $\mathcal{M}$. At the start of round t , write the state of $\mathcal{M}$ as

$$\mathbf{s}_t^{\mathcal{M}} = (\chi_t, \text{Hist}_t, \mathbf{h}_t, \pi_t),$$

where χ_t is the environment state, Hist_t is the ordered message log, $\mathbf{h}_t$ collects all private module states and memories, and π_t is the current pointer into ω . Each message has the form

$$m = (\text{id}, \text{src}, \text{dst}, \tau, \text{payload}, \text{ts}), \quad \tau \in \mathcal{S}_{\mathcal{M}},$$

where $\mathcal{S}_{\mathcal{M}}$ is finite by typed message passing. Let

$$\text{Vis}_t(\text{Hist}) = \{m \in \text{Hist} : m.\text{ts} \leq t\}$$

denote the PIT-visible sub-log, preserving the bus order inherited from Hist . The PIT assumption says that every activation at round t is a function only of its PIT-visible inbox, PIT data $\mathcal{D}_t$, the relevant component state, and the next random tape segment. The stage-separation assumption says that each terminated round is organized into evidence ingestion, fusion/scoring, risk/constraints, and execution phases. Any finite internal debate, reflection, contest, or tool-call sequence inside a phase is treated as a sequence of micro-steps. If the original implementation uses adaptive branching, the branch rule is included in the activated component; inactive branches emit no messages and consume no randomness. Equivalently, one may use a finite super-schedule that calls all possible phase components in a fixed order, with no-ops on branches not taken.

Construction of G . We define the GMATS instance as follows. Set the asset universe $\mathcal{A}$, data stream $\mathcal{D} = \{\mathcal{D}_t\}_{t \in T}$, and execution environment $\mathcal{E}$ equal to those of $\mathcal{M}$. Set the message schema to

$$\mathcal{S} = \mathcal{S}_{\mathcal{M}} \cup \{\text{STANCE}, \text{DRAFTORDER}, \text{APPROVEDORDER}\},$$

where the three extra types are wrappers used only when $\mathcal{M}$'s native artifacts do not already distinguish stances, draft orders, and approved orders. The payloads of all native messages are preserved verbatim; the wrappers do not alter the information available to any agent.

Let $\mathcal{U}$ be the set of components of $\mathcal{M}$. For each component $u \in \mathcal{U}$ and each phase $\phi \in \{\text{ingest}, \text{fuse}, \text{risk}, \text{exec}\}$ in which u participates, create a phase-restricted GMATS component u^ϕ . The transition function of u^ϕ is exactly the transition function used by u in phase ϕ of $\mathcal{M}$: on the same PIT-visible inbox, PIT data, private state, and random tape segment, it appends the same messages, produces the same local outputs, and updates the same private state. If u spans several phases, the corresponding phase-restricted components share the same private state cell, so splitting u across stages does not change its memory. Fresh message identifiers, counters, and tool-call caches are included in this private state or in ω .

Assign the phase-restricted components to GMATS stages by

$$A = \{u^{\text{ingest}}\}, \quad C = \{u^{\text{fuse}}\}, \quad K = \{u^{\text{risk}}\}, \quad X = \{u^{\text{exec}}\}.$$

If $\mathcal{M}$ has no separate risk or execution module, the corresponding set is empty and the adjacent stage emits the final order, as allowed by GMATS. Define Σ

to activate these components in the same within-round order as $\mathcal{M}$'s phases and micro-steps, using the same random tape offsets. Finally, define $\mathcal{L}_t$ and $\mathcal{A}_t$ as the aggregate input–output maps induced by the coordinator and controller micro-steps scheduled at round t . Thus GMATS exposes the same staged interface as Definition 1 while preserving the original component-level behavior.

Correctness Invariant. Let

$$\mathbf{s}_t^G = (\chi_t^G, \text{Hist}_t^G, \mathbf{h}_t^G, \pi_t^G)$$

be the analogous GMATS state. We prove, by induction over rounds, the stronger invariant

$$\mathcal{I}(t) : \begin{array}{ll} \chi_t^{\mathcal{M}} = \chi_t^G, & \text{Hist}_t^{\mathcal{M}} = \text{Hist}_t^G \quad \text{as ordered logs,} \\ \mathbf{h}_t^{\mathcal{M}} = \mathbf{h}_t^G, & \pi_t^{\mathcal{M}} = \pi_t^G. \end{array}$$

This implies equality of PIT-visible inboxes for every component at round t .

Base Case. At $t = 1$, the two systems have the same initial environment state, the same initial log (typically empty), the same component states by construction, and the same random-tape pointer. Therefore $\mathcal{I}(1)$ holds.

Inductive Step. Assume $\mathcal{I}(t)$ holds. Consider the micro-steps of round t in their scheduled order. Before the first micro-step, all PIT-visible inboxes, PIT data, private states, and random-tape positions agree. Suppose they agree before some micro-step. The activated component of $\mathcal{M}$ and the corresponding phase-restricted component of G then apply the same transition function to the same inputs and the same random tape segment. They therefore append the same sequence of messages to the bus, update the same private states, produce the same local output, and advance the random pointer by the same amount. Thus equality is preserved after that micro-step. A nested induction over all micro-steps in the round gives equality of the full ordered logs, private states, and random pointers at the end of the fusion and risk phases. In particular, both systems obtain the same approved order o'_t or the same no-trade decision.

The execution environments are identical and are called with the same input state and order, hence

$$(\text{fills}_t, \chi_{t+1}) = \mathcal{E}(o'_t, \chi_t)$$

produces the same fills and next environment state in both systems. Messages with future timestamps, if any, have also been appended identically, so the full ordered logs remain equal and the PIT-visible logs will agree at the next round. Consequently $\mathcal{I}(t + 1)$ holds.

Conclusion. By induction, $\mathcal{I}(t)$ holds for every round. Therefore the order submitted to the execution environment is identical at every decision time. The GMATS instance G thus produces the same executed-order sequence as $\mathcal{M}$ under the same initial state and randomness.

Algorithm 1 Compile a staged multi-agent pipeline into GMATS

-
- 1: **Input:** Pipeline $\mathcal{M}$ with agents/modules $\mathcal{U}$, PIT data $\{\mathcal{D}_t\}$, environment $\mathcal{E}$, and within-round micro-schedule $\Sigma_{\mathcal{M}}$.
 - 2: **Output:** GMATS instance G with identical executed-order sequence under matched randomness.
 - 3: Set $\mathcal{A}$, $\{\mathcal{D}_t\}$, and $\mathcal{E}$ to match $\mathcal{M}$.
 - 4: Define a finite schema $\mathcal{S}$ by wrapping each artifact class in $\mathcal{M}$ (reports, debate turns, candidate orders, risk checks) as a typed message; optionally add STANCE, DRAFTORDER, APPROVEDORDER.
 - 5: Partition computations into phases (i)–(iv) and create phase-restricted agents u^ϕ for each $u \in \mathcal{U}$ and phase ϕ .
 - 6: Assign $u^{(i)}$ to A , $u^{(ii)}$ to C , $u^{(iii)}$ to K , and $u^{(iv)}$ to X ; use an empty stage when the original pipeline has no separate module.
 - 7: Define Σ by unrolling $\Sigma_{\mathcal{M}}$ each round; debates, contests, and reflection become micro-rounds inside A , C , K , or X .
 - 8: Define $\mathcal{L}_t$ and $\mathcal{A}_t$ as the maps induced by running the coordinator and controller micro-rounds under Σ .
 - 9: **return** G .
-

2 Compiling Modern Agentic Trading Frameworks into GMATS

This section expands Corollary 1 from the main paper by giving a concrete *compilation* procedure from representative multi-agent trading frameworks into a GMATS instance. The key idea is that most modern LLM trading systems already implement a firm-like pipeline: (i) evidence ingestion, (ii) multi-signal fusion and scoring, (iii) risk/constraints checks, and (iv) execution. GMATS makes this structure explicit via the $A \rightarrow C \rightarrow K \rightarrow X$ staging and a schedule Σ that unrolls internal debates, contests, and reflections as micro-rounds.

2.1 A Generic Compilation Recipe

Let $\mathcal{M}$ be any multi-agent pipeline satisfying Theorem 1. A *GMATS compiler* produces a tuple $G = (\mathcal{A}, A, C, K, X, \mathcal{S}, \Sigma, \mathcal{D}, \mathcal{L}, \mathcal{A}, \mathcal{E})$ as follows.

Typed Message Passing. Many agentic frameworks use free-form natural-language artifacts internally, such as chat logs, debate turns, or reflective notes. GMATS requires only a finite schema, so compilation can introduce lightweight wrappers such as FUNDREPORT, BEARARG, RISKCHECK, and CANDIDATE-ORDER while preserving payloads verbatim. This does not change behavior as long as the routing and inbox functions respect the wrapper types.

PIT Discipline. The GMATS instance inherits PIT discipline from the evaluation harness. At round t , the compiler provides only $\mathcal{D}_t$ and messages with timestamps at most t to all agents. If the original framework has asynchronous tool calls, the compiler treats tool outputs as environment data with explicit timestamps and delays delivery until the appropriate round.

2.2 Case Study: TradingAgents $\Rightarrow$ GMATS

TradingAgents [3] is organized as a trading-firm simulation with (i) an Analyst Team (fundamental, sentiment, news, technical), (ii) a Researcher Team with bull and bear researchers that debate the analyst outputs, (iii) a Trader Agent that synthesizes a transaction proposal, and (iv) a risk-management and portfolio-manager gate that approves or rejects before simulated execution.

A semantics-preserving mapping into GMATS is:

- A : {Fundamentals Analyst, Sentiment Analyst, News Analyst, Technical Analyst} emitting typed insight messages, one per role per round.
- C : {Bull Researcher, Bear Researcher, Trader}. The bull/bear debate is unrolled as micro-rounds in Σ producing debate-turn messages; the Trader consumes the final debate state and emits a DRAFTORDER or stance vector.
- K : {Risk Management Team, Portfolio Manager}. This stage produces APPROVEDORDER or a reject decision; limits such as exposure, leverage, and liquidity live in Λ_t .
- X : the simulated exchange/execution module submitting o'_t to $\mathcal{E}$ and logging fills.

2.3 Case Study: TradingGroup $\Rightarrow$ GMATS

TradingGroup [2] consists of specialized signal agents, including news sentiment, financial-report interpretation, trend forecasting, and style adaptation, plus a decision agent that merges signals. It also includes self-reflection and a dynamic risk model with stop-loss/take-profit logic.

A GMATS mapping is:

- A : {Sentiment, Report, Trend, Style} agents emitting typed signals.
- C : a decision-making coordinator that fuses signals into a stance or order. Self-reflection is compiled into coordinator micro-rounds that write/read a memory state, still under PIT because memory is generated only from past rounds.
- K : the dynamic risk-management module enforcing stop-loss/take-profit and exposure caps as Λ_t .
- X : the execution interface submitting approved orders to $\mathcal{E}$.

2.4 Other Examples: ContestTrade and QuantAgent

ContestTrade [5] decomposes into a Data Team that compresses raw data into text factors and a Research Team that proposes candidate decisions. Its internal contest mechanism can be compiled as coordinator micro-rounds that score and select candidate orders before risk checks.

QuantAgent [4] decomposes signal extraction into specialized technical agents, such as Indicator, Pattern, and Trend agents, with a Risk agent. These map naturally to analyst and controller roles, with a coordinator that fuses the technical messages into a final stance or order. In both cases, the contest/reflection logic is represented as micro-rounds in Σ and does not require any extension to GMATS beyond typed messages and scheduling.

Table 1. Example mappings from agentic frameworks to GMATS stages.

Framework	Analysts (A)	Coordinators (C)	Controllers/Exec (K/X)
TradingAgents	role analysts	bull/bear debate, trader	risk/PM, exchange
TradingGroup	signal agents	decision + reflection	dynamic risk, executor
ContestTrade	data team	research contest	risk gate, executor
QuantAgent	indicator, pattern, trend	fusion decision	risk, executor

3 Additional Experiment Results: TradingAgents Baseline

This section reports an additional Always Negative attacker result for the *TradingAgents* [3] baseline. The protocol is the same attack-clean protocol used in the main paper. All entries are averaged over 5 runs and are reported as attack minus clean. As in the main paper, we emphasize deltas rather than absolute Sharpe levels, since Sharpe can be unstable over short horizons [1].

Table 2. *TradingAgents* baseline under the Always Negative attacker. Entries are attack-clean deltas averaged over 5 runs.

Metric	15%	30%	45%	60%	75%
BSS_{analyst}	-0.788	-1.048	-1.261	-1.243	-1.369
BSS_{bull}	-0.309	-0.302	-0.279	-0.293	-0.250
BSS_{bear}	-0.019	-0.032	-0.029	-0.028	-0.025
BSS_{fuse}	-0.247	-0.241	-0.193	-0.210	-0.152
ΔCR	-0.021	-0.020	-0.021	-0.021	-0.017
ΔSharpe	+7.764	+5.004	+1.226	+2.018	+1.942
ΔVol	-0.030	-0.022	-0.020	-0.019	-0.015

TradingAgents exhibits the same directional social contagion observed in GMATS. The analyst belief shift grows in magnitude with poisoning: BSS_{analyst} decreases from -0.788 at 15% poisoning to -1.369 at 75% poisoning. However, contagion is damped downstream. The fused belief shift remains modest and becomes less negative at higher poisoning, moving from $BSS_{\text{fuse}} = -0.247$ at 15% to $BSS_{\text{fuse}} = -0.152$ at 75%. This is consistent with partial cancellation or regularization at the coordinator stage rather than full propagation of the contaminated social signal.

On performance, TradingAgents becomes more conservative under attack: ΔVol is consistently negative, while ΔCR is mildly negative. Over this short window, lower realized volatility can mechanically yield positive ΔSharpe even

when return deltas are slightly negative. Accordingly, we interpret this baseline as risk-damping under attack rather than as evidence that poisoning improves profitability.

Acknowledgments. This research/project is supported by the National Research Foundation, Singapore under its AI Singapore Programme (AISG Award No: AISG3-PhD/2025-08-063).

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

Generative AI Assistance Disclosure. We used a large language model as a writing aid to improve readability (e.g., grammar, spelling, and wording suggestions). The model was not used to generate new scientific content, results, proofs, or experimental data, nor to make decisions about the research contributions. All text was reviewed and edited by the authors, who take full responsibility for the final manuscript and for ensuring compliance with applicable research integrity and copyright requirements.

References

1. Lo, A.W.: The statistics of sharpe ratios. *Financial Analysts Journal* **58**(4), 36–52 (2002). <https://doi.org/10.2469/faj.v58.n4.2453>
2. Tian, F., Salim, F.D., Xue, H.: Tradinggroup: A multi-agent trading system with self-reflection and data-synthesis (2025). <https://doi.org/10.48550/arXiv.2508.17565>, <https://arxiv.org/abs/2508.17565>, arXiv:2508.17565 (v1), 2025-08-25
3. Xiao, Y., Sun, E., Luo, D., Wang, W.: Tradingagents: Multi-agents llm financial trading framework (2025). <https://doi.org/10.48550/arXiv.2412.20138>, <https://arxiv.org/abs/2412.20138>, arXiv:2412.20138 (v7), last revised 2025-06-03
4. Xiong, F., Zhang, X., Feng, A., Sun, S., You, C.: Quantagent: Price-driven multi-agent llms for high-frequency trading (2025). <https://doi.org/10.48550/arXiv.2509.09995>, <https://arxiv.org/abs/2509.09995>, arXiv:2509.09995 (v3), last revised 2025-09-27
5. Zhao, L., Sun, R., Jiang, Z., Yang, B., Bai, Y., Chen, M., Wang, X., Li, J., Bai, Z.: Contesttrade: A multi-agent trading system based on internal contest mechanism (2025). <https://doi.org/10.48550/arXiv.2508.00554>, <https://arxiv.org/abs/2508.00554>, arXiv:2508.00554 (v3), last revised 2025-08-18