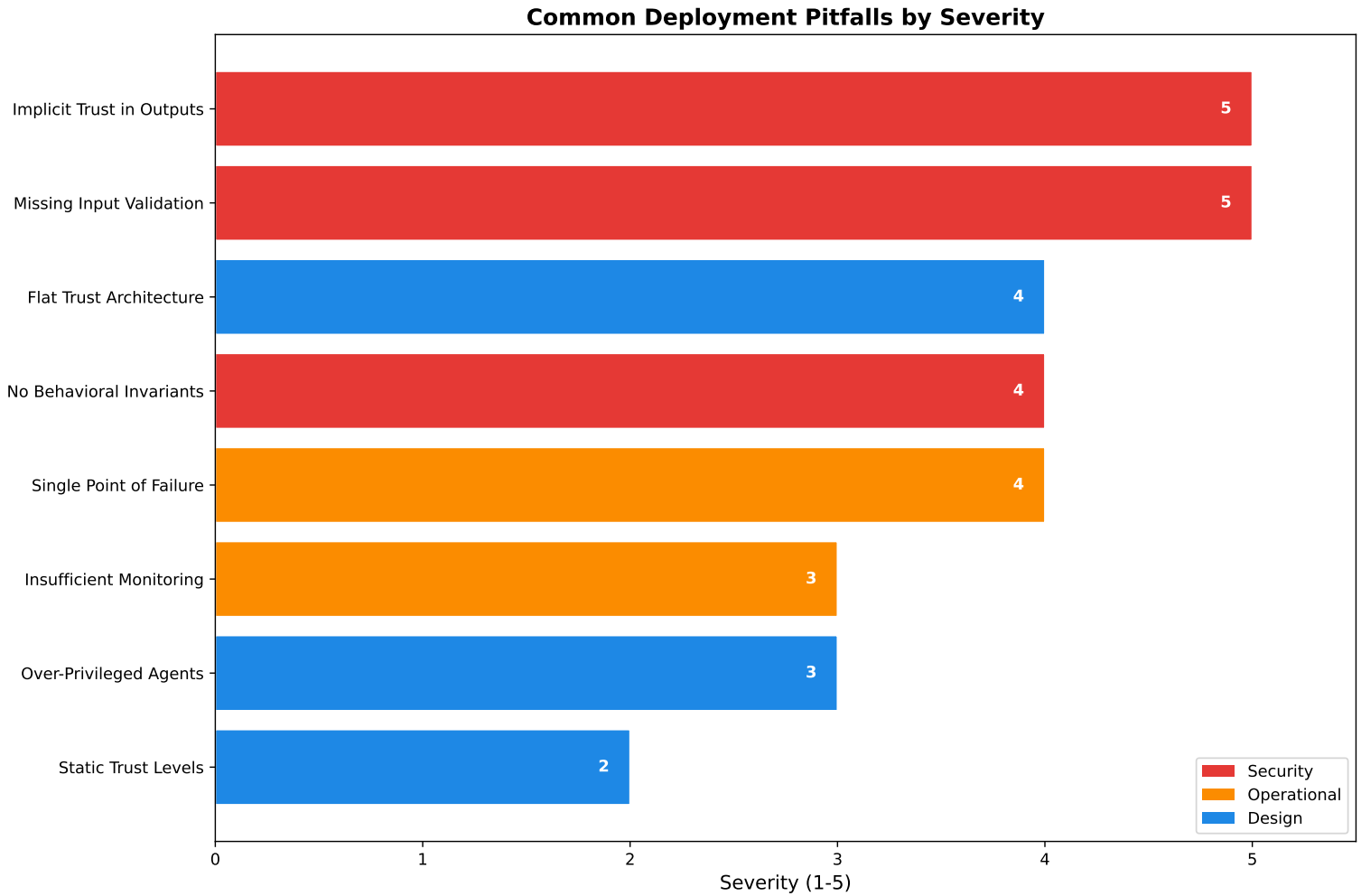
Common Pitfalls

This section documents anti-patterns observed in multiagent deployments. Each entry describes the pattern, its consequences, and specific mitigations. Figure 1 ranks these pitfalls by severity to guide remediation prioritization.



Pitfall 1: Implicit Trust

Pattern: Treating all inter-agent communication as trusted by default.

Indicators: - No source verification on agent messages - All agents have equal authority regardless of role - Delegation without bounds or decay

Consequences: - Single compromised agent influences entire system - Trust amplification attacks succeed (see Part 1, Section 3.4) - No containment of adversarial content

Mitigation: 1. Implement explicit trust scoring on inter-agent channels 2. Require minimum trust thresholds for consequential actions 3. Apply delegation decay (< 1 per hop) 4. Verify source on every inter-agent message

Pitfall 2: Security as Afterthought

Pattern: Adding cognitive security after architecture is finalized.
Indicators: - Security checks only at external interfaces - Core agent logic has no security awareness - Belief provenance untracked
Consequences: - Bypass opportunities at integration points - Performance overhead from external security layers - Incomplete attack surface coverage
Mitigation: 1. Design cognitive security into architecture from the start 2. Embed trust checks in delegation logic 3. Build provenance tracking into belief management 4. Include security constraints in agent system prompts

Pitfall 3: Uncalibrated Thresholds

Pattern: Setting security thresholds without understanding tradeoffs.

Indicators: - Thresholds copied from examples without adjustment - Same thresholds for all contexts - No testing against representative attacks

Consequences: - Too strict: high false positive rate, user friction - Too permissive: attacks succeed undetected - Settings mismatched to actual risk profile

Mitigation: 1. Assess risk profile before configuring (see Section 6) 2. Test thresholds against representative attack samples (Part 2 corpus) 3. Monitor false positive/negative rates in production 4. Adjust based on operational feedback

Pitfall 4: Individual-Only Security

Pattern: Focusing on single-agent security while ignoring multi-agent attack surfaces.
Indicators: - No consensus mechanism for critical decisions - Agent count changes without verification - No Sybil resistance
Consequences: - Fake agents influence collective decisions - Consensus manipulation - Coordination failures masked as normal disagreement
Mitigation: 1. Implement Byzantine consensus for critical collective decisions 2. Require agent authentication before vote counting 3. Monitor for unusual coordination patterns 4. Apply quorum requirements assuming adversarial presence
Part 1, Section 4.5 formalizes Byzantine consensus requirements.

Pitfall 5: Static Tripwires

Pattern: Deploying canary tripwires once without rotation.
Indicators: - Same canary values since deployment - No rotation schedule - Predictable canary locations
Consequences: - Sophisticated adversaries learn to avoid canaries - Effectiveness degrades over time - False confidence in detection coverage
Mitigation: 1. Implement automated canary rotation 2. Vary placement across agents and belief categories 3. Monitor canary check patterns, not just modifications 4. Include non-obvious canaries

Pitfall 6: Ignoring Progressive Drift

Pattern: Only alerting on large, sudden belief changes.
Indicators: - High threshold for drift alerts - No long-term drift tracking - Static baseline
Consequences: - Sub-threshold changes accumulate undetected - Beliefs slowly corrupted - Significant deviation without alert
Mitigation: 1. Use sliding window drift detection 2. Track cumulative drift, not just per-update delta 3. Periodic baseline comparison 4. Alert on trend as well as absolute magnitude
Part 1, Definition 5.1 formalizes drift scoring.

Pitfall 7: Insufficient Logging

Pattern: Retaining insufficient information for post-incident analysis.
Indicators: - Only final decisions logged - No belief state history - Inter-agent messages disposed after processing
Consequences: - Cannot reconstruct attack path - Cannot identify injection point - Cannot assess full impact scope
Mitigation: 1. Log all belief updates with provenance tags 2. Retain inter-agent message history 3. Periodic cognitive state snapshots 4. Structured logging for causal analysis

Pitfall 8: Single-Orchestrator Reliance

Pattern: Relying entirely on orchestrator integrity without backup.
Indicators: - Single orchestrator for entire system - No orchestrator monitoring - Workers unconditionally trust orchestrator
Consequences: - Orchestrator compromise = total system compromise - No recovery path - Complete trust inversion attack possible (see Part 1, Section 2.3)
Mitigation: 1. Consider multi-orchestrator architectures for critical decisions 2. Monitor orchestrator behavior with same rigor as agents 3. Workers verify orchestrator identity on critical commands 4. Implement orchestrator-specific tripwires

Summary Checklist

Pitfall	Assessment	Status
Implicit trust	Trust scoring implemented?	
Security afterthought	Security in initial architecture?	
Uncalibrated thresholds	Thresholds tested against attacks?	
Individual-only security	Byzantine consensus deployed?	
Static tripwires	Canary rotation scheduled?	
Ignoring drift	Progressive drift monitoring?	
Insufficient logging	Full belief history retained?	
Single orchestrator	Orchestrator monitored?	

Address unchecked items before production deployment.